

Dr. Dobb's Journal of

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

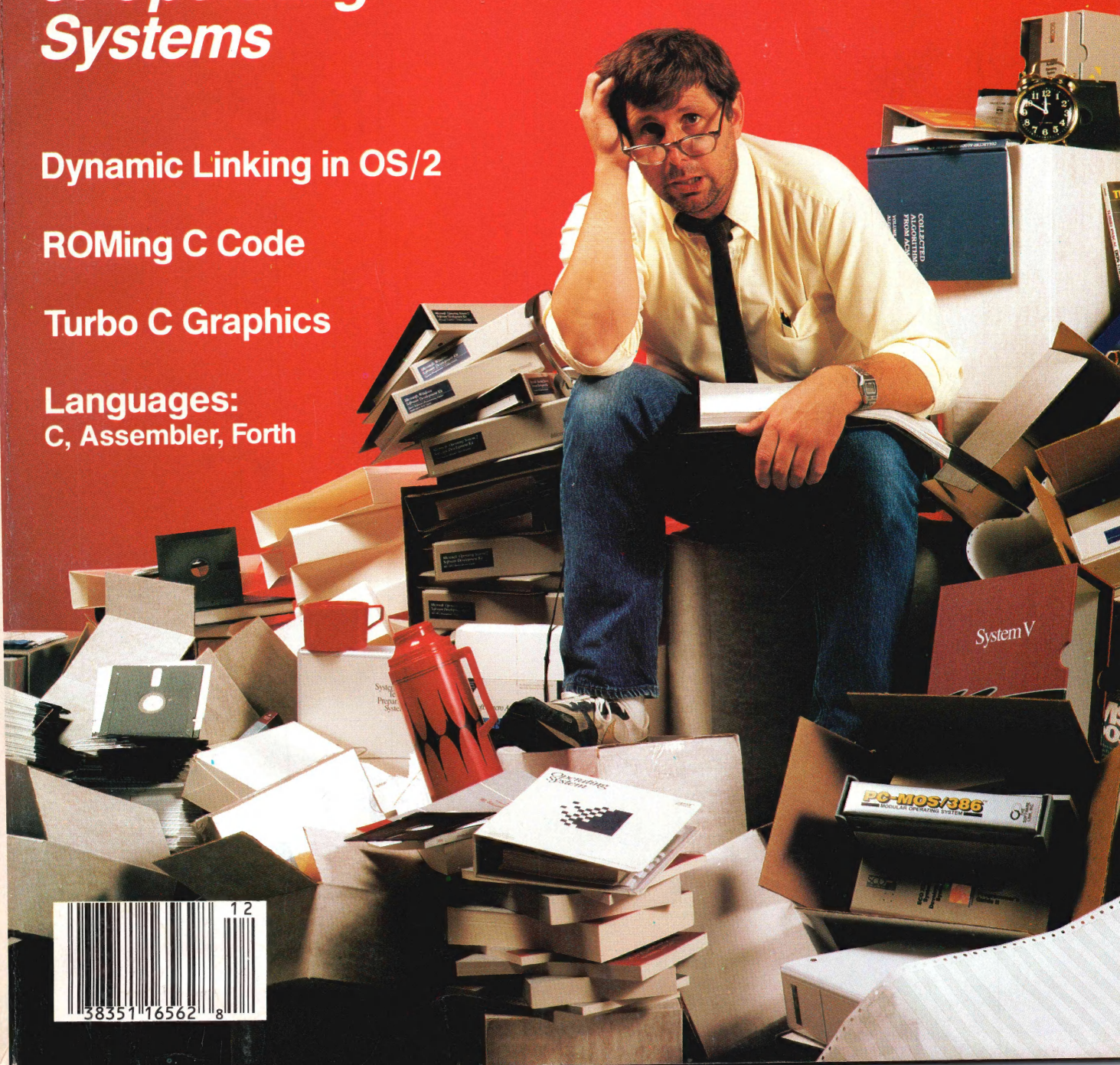
Making Sense of Operating Systems

Dynamic Linking in OS/2

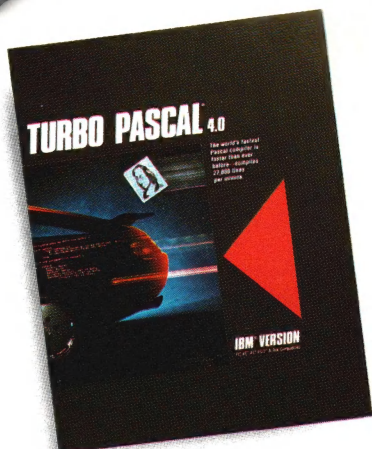
ROMing C Code

Turbo C Graphics

Languages:
C, Assembler, Forth



man ever before!



4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately—so that an error in one module is seeable and fixable, and you're not sent through all your code to find one error. Compiling and linking these separate units happens in a

flash because your compiling horsepower is better than 27,000 lines a minute.* And 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message at the top of your screen *and* the cursor flags the error's location for you.

4.0 gives you an integrated programming environment

4.0's integrated environment includes pull-down menus and a built-in editor. Your program output is

automatically saved and shown in the output window. You can Scroll, Pan, or Page through all your output and know where everything is all the time. Given 4.0's integration, you can edit, compile, find and correct errors—all from inside the integrated development environment.

You'll never lose your mind, because 4.0 never loses your place

Whenever you re-load 4.0, it remembers what you and it were doing before you left. It puts you right back in the editor with the same file and in the same place as you were working last.

*Run on an 8 MHz IBM AT.

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
Copyright © 1987 Borland International, Inc. BI 1159

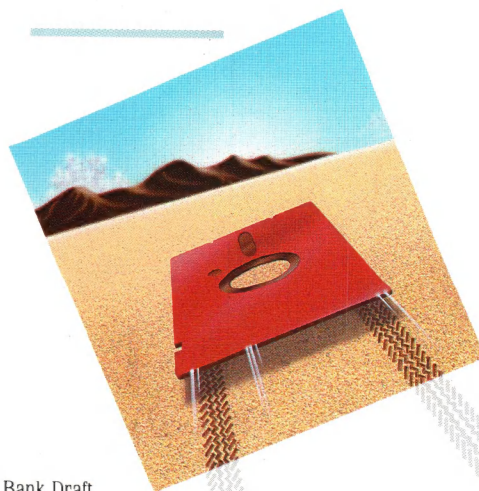
Please check box(es)	Sugg. Retail	Upgrade Price†	Serial No.
<input type="checkbox"/> Turbo Pascal 4.0 Compiler	\$ 99.95	\$ 39.95	_____
<input type="checkbox"/> Turbo Pascal 4.0 Developer's Library	395.00	150.00	_____
(Includes Turbo Pascal Tutor and all Toolboxes; must be ordered with Compiler)			
<input type="checkbox"/> Turbo Pascal Tutor	69.95	19.95	_____
<input type="checkbox"/> Turbo Pascal Database Toolbox	99.95	29.95	_____
<input type="checkbox"/> Turbo Pascal Graphix Toolbox	99.95	29.95	_____
<input type="checkbox"/> Turbo Pascal Editor Toolbox	99.95	29.95	_____
<input type="checkbox"/> Turbo Pascal Numerical Methods Toolbox	99.95	29.95	_____
<input type="checkbox"/> Turbo Pascal Gameworks	99.95	29.95	_____
Total product amount	\$ _____		
CA and MA residents add sales tax	\$ _____		
Shipping and handling*	\$ _____		
Total amount enclosed	\$ _____		

Please specify diskette size: ☐ 5¼" ☐ 3½"

Credit card expiration date: _____/_____/_____

Card # _____

Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft



*In US please add \$5 shipping for each product ordered or \$15 for the Compiler and Developer's Library. Outside US please add \$10 shipping and handling for each product ordered or \$25 for the Compiler and Developer's Library.

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.



```
record used by Intr and MSdos )  
= record  
  case Integer of  
    0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Word;  
       1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);  
  end;  
e and untyped-file record )  
record  
  Handle: Word;  
  Mode: Word;  
  RecSize: Word;  
  Private: array[1..26] of Byte;  
  UserData: array[1..16] of Byte;  
  Name: array[0..79] of Char;
```

Program in the
fast lane with
Borland's new
Turbo Pascal 4.0.

The fast lane is *fast*

Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute. That's much faster than 3.0 or any other Pascal compiler and the reason why you need 4.0 today.

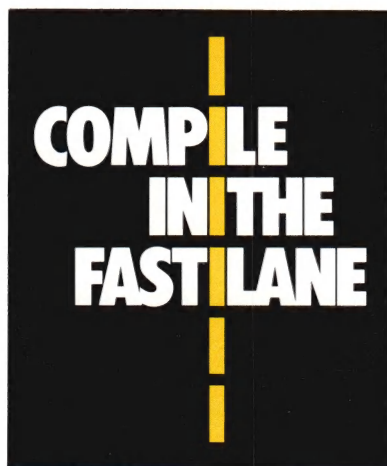
Pascal. The fastest and the best.

If you're just now learning a computer language, learn Pascal. If you're already programming in Pascal, you're programming with a winner because Pascal is the worldwide language of choice. Pascal is the most popular language in university computer science classes and with computer enthusiasts who appreciate Pascal's modern programming

structure. It's powerful, coherent, easy to learn and use—and with Turbo Pascal 4.0—faster than ever before.

Turbo Pascal: Technical excellence

Commitment to technical excellence and



superiority also means commitment to detail, however painstaking, and that takes time. 4.0's pre-

decessor, Turbo Pascal 3.0 is the worldwide standard, and with Turbo Pascal 4.0, we've bettered that standard. 4.0 is clearly the world's fastest development tool for the IBM® PS/2 series, PC's and compatibles—and the world's favorite Pascal compiler.

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use every byte of memory in your computer. You paid for all that memory, now you can use it freely.

For the IBM PS/2 and the IBM and Compaq families of personal computers and all 100% compatibles.

YES!

I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes

Registered owners have been notified by mail. If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at (800) 543-7543. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

**Pascal 4.0 Upgrade Dept.
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066**

Name _____
Ship Address _____
City _____ State _____
Zip _____ Telephone () _____

This offer is limited to one upgrade per valid registered product. It is good until November 30, 1987. Not good with any other offer from Borland. Please allow 4 to 6 weeks for delivery of Toolboxes.

Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

Now's the time for a *fast* decision: Upgrade now to 4.0!

Compatibility with Turbo Pascal 3.0

We've created 4.0 to be highly compatible with version 3.0 and included a conversion program and compatibility units to help you convert all your 3.0 programs to 4.0.

Highlights of Borland's new Turbo Pascal 4.0

- Compiles 27,000 lines per minute
- Supports >64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler

4.0 also

- Saves output screen in a window
- Supports 25, 43 and 50 lines per screen
- Generates MAP files for debugging
- Has graph units including CGA, EGA, VGA, MCGA, 3270 PC, AT & T 6300 & Hercules support
- Supports extended data types (including word, long integers)
- Does smart linking
- Comes with a free revised MicroCalc spreadsheet source code

4.0 is all yours for only \$99.95

Sieve (25 iterations)

	Turbo Pascal 4.0	Turbo Pascal 3.0
<i>Size of Executable File</i>	2224 bytes	11682 bytes
<i>Execution speed</i>	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our GOMOKU program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of GO.PAS (1006 lines)

	Turbo Pascal 4.0	Turbo Pascal 3.0
<i>Compilation speed</i>	2.2 seconds	3.6 seconds
<i>Lines per minute</i>	27,436	16,750

GO.PAS compiled on an 8 MHz IBM AT

60-Day Money-Back Guarantee**



*For the dealer nearest
you or to order call*

(800) 543-7543.

CIRCLE 101 ON READER SERVICE CARD

BORLAND

Blaise puts the Accent on C with C TOOLS PLUS/5.0™

Enhance your Microsoft C programming environment with C TOOLS PLUS/5.0™—a new, quintessential library of C functions. C TOOLS PLUS/5.0 from Blaise Computing Inc. puts a prime accent on quickly building professional applications using the full power of Microsoft C Version 5.0 and QuickC. Now you can concentrate on program creativity by having full control over DOS, menus, interrupt service routines, memory resident programs, printer and keyboard control, and more!

C TOOLS PLUS/5.0 prebuilt libraries are ready to use with either QuickC or the Microsoft C Version 5.0 command line environment. Complete documented source code is included so that you can study and adapt it to your specific needs. Blaise Computing's attention to detail, like the use of full function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes C TOOLS PLUS/5.0 the choice for experienced developers as well as newcomers to C.

Continuous refinement of Blaise Computing's library products has produced a collection of tools that are unsurpassed for reliability, functionality and ease of use. Built upon the widely acclaimed C TOOLS PLUS, C TOOLS PLUS/5.0 includes such highly-developed features as:

◆WINDOWS

- Stackable, removable.
- Optional borders, cursor memory.
- Accept user input, formatted output.
- "printf" window-oriented output. **NEW!**

◆INTERRUPT SERVICE ROUTINES

- Capture DOS critical errors and keystrokes.
- Install hardware interrupt handlers.

◆RESIDENT SOFTWARE SUPPORT

- Install, detect and remove memory resident programs.

◆MENUS

- Horizontal and pulldown. **NEW!**
- Lotus-style support. **NEW!**

◆INTERVENTION CODE

- Schedule C functions at specified times, intervals or with a "hot key." **NEW!**
- Take full advantage of DOS, even from memory resident programs. **NEW!**

◆FAST DIRECT VIDEO ACCESS

- All monitors, even EGA 43-line mode.

◆PRINTER CONTROL

- Access BIOS print functions. **NEW!**
- Control the DOS PRINT utility. **NEW!**

◆UTILITIES AND MACROS

- Take advantage of DOS file structure.
- Manipulate data types, far & near pointers. **NEW!**
- Access any memory areas with fast "peek" and "poke" macros. **NEW!**

C TOOLS PLUS/5.0 supports the Microsoft C Version 5.0 and QuickC compilers, requires DOS 2.00 or later and is just **\$129.00**.

C ASYNCH MANAGER™ Version 2.0 IMPROVED!

C ASYNCH MANAGER is a library of functions designed to help you incorporate asynchronous communication capabilities into your application programs. Version 2.0 has been rewritten especially for Microsoft C Version 5.0 and Borland's Turbo C. Simultaneous buffered input and output to both COM ports at speeds up to 9600 baud, XON/XOFF protocol, modem control and XMODEM file transfer are among the many features supported and is priced at just **\$175.00**.

Blaise computing Inc. has a full line of support products for both Pascal and C. Call today for your free information packet.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 102 ON READER SERVICE CARD

Now, just \$129 and supports Microsoft C 5.0 and QuickC

THE BLAISE MENU

VIEW MANAGER

\$275.00

General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

Turbo C TOOLS

\$129.00

Windows; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. For Turbo C.

Turbo POWER SCREEN

COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo POWER TOOLS PLUS

\$129.00

Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. Now supports Turbo Pascal 4.0!

Turbo ASYNCH PLUS

\$129.00

Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. Now supports Turbo Pascal 4.0!

PASCAL TOOLS/TOOLS 2

\$175.00

Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

ASYNCH MANAGER

\$175.00

Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For MS-Pascal.

KeyPlayer

\$49.95

"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC

\$95.00

NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF

\$49.95

Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

LIGHT TOOLS

\$99.95

Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

TO ORDER CALL TOLL FREE

800-333-8087

TELEX NUMBER-338139

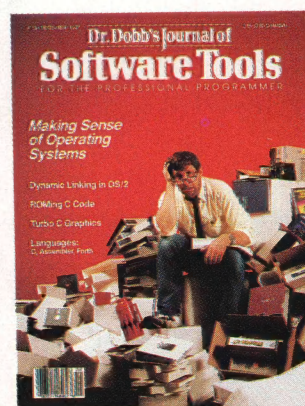
Yes! Send me the prime accent!

Enclosed is \$_____ for _____ copies of _____
☐ Please send me more information on your products.
 CA residents add Sales Tax. Domestic orders add \$4.00 for UPS shipping, \$10.00 for Federal Express standard air.
 Name: _____ Phone: (____) _____
 Address: _____ State: _____ Zip: _____
 City: _____ Exp. Date: _____
 VISA or MC#: _____

Microsoft is a registered trademark of Microsoft Corporation. QuickC is a trademark of Microsoft Corporation. Turbo C is a registered trademark of Borland International.

ARTICLES

- OS/2** ► **Dynamic Linking in OS/2** **18**
by D. E. Cortesi
 Dave explains one of the less explored aspects of OS/2—the built-in hooks for third-party extensions.
- RAM-Caching** ► **A RAM-Cache Manager in C** **30**
by Alan Deikman
 Alan provides a set of caching routines for optimizing program performance.
- EXE to ROM** ► **Putting ROM Code in its Place** **38**
by Rick Naro
 Rick's LOCATE utility lets you move code from DOS .EXE files to ROM environments.
- Faster Roots** ► **Integers Don't Float** **48**
by Ray Mariella
 Ray investigates the tradeoff between speed and precision in some algorithms used for calculating square roots.
- Text Graphics** ► **A Graphics Toolbox for Turbo C—Part II** **54**
by Kent Porter
 Kent builds upon the library he offered last issue. This time the subject is text graphics: items like menu bars, pop-up windows, and pull-down menus.



About the Cover

We'd like to be able to tell you that you could avoid all this OS documentation just by reading this single issue of DDJ. We'd like to, but we can't. The new trend in operating systems seems to be an entry requirement of at least six months' study.

This Issue

It's the return of DDJ's infamous annual Operating Systems issue. So why are you reading this instead of the articles listed on the left?

Next Issue

January is DDJ's annual 68K issue, and we're starting the new year off right with a new column devoted to advanced Mac hacking, er, programming. Plus a couple of other surprises.

COLUMNS

- Turbo C** ► **C CHEST** **126**
by Allen Holub
 Allen presents a multitasking kernel that allows a program to run several subroutines as independent tasks.
- Forth** ► **THE FORTH COLUMN** **144**
by Martin Tracy
 News from the August ANSI Forth meeting as well as a set of Forth-83 words to extend string handling capabilities.

FORUM

- EDITORIAL** **6**
by Tyler Sperry
- RUNNING LIGHT** **8**
by Tyler Sperry
- ARCHIVES** **8**
- LETTERS** **10**
by you

PROGRAMMER'S SERVICES

- ADVERTISER INDEX:** **153**
 Where to go for more information on products.
- OF INTEREST** **154**
 Products for programmers
- SWAINE'S FLAMES** **160**
by Michael Swaine



Peter Norton. new programmi who hate

THE NORTON *On-Line Programmer's* GUIDES™

The ultimate productivity tool for programmers. ■ Puts volumes of cross-referenced data at your fingertips. ■ Replaces most manual searches with a few simple keystrokes. ■ Includes compiler for creating your own databases. ■ Also available in versions for BASIC, C and Pascal.

ASSEMBLY



For the complete IBM® PC family and compatibles.

Nobody ever said programming PCs was supposed to be easy.

But does it have to be tedious and time-consuming, too?

Not any more.

Not since the arrival of the remarkable new program on the left.

Which is designed to save you most of the time you're currently spending searching through the books and manuals on the shelf above.

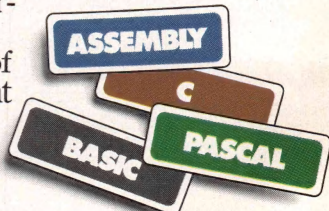
The Norton On-Line Programmer's Guides™ are a quartet of pop-up reference packages that do the same things in four different languages.

Each package consists of two parts: A memory-resident Instant Access™ program. And a comprehensive, cross-referenced database crammed with just about everything you need to know to program in your favorite language.

And when we say everything, we mean everything.

Everything from information about language

Designed for IBM® PC, PC-AT and DOS compatibles. Available at most software





announces a ng tool for people manual labor.

syntax to a variety of tables, including ASCII characters, line drawing characters, error messages, memory usage maps, important data structures and more.

How much more?

Well, the databases for BASIC, C and Pascal give you detailed listings of all built-in and library functions.

While the Assembly database delivers a complete collection of DOS service calls, interrupts and ROM BIOS routines.

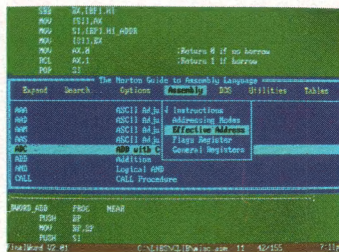
You can, of course, find most of this information in the books and manuals on our shelf.

But Peter Norton—who's written a few books himself—figured you'd rather have it on your screen.

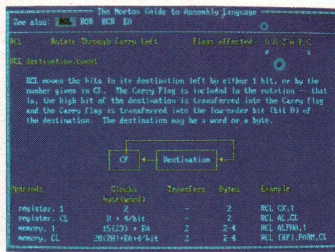
In seconds.

In full-screen or moveable half-screen mode.

Popping up right next to your work. Right where you need it.



A Guides reference summary screen (shown in blue) pops up on top of the program you're working on (shown in green).



Summary data expands on command into extensive detail. And you can select from a wide variety of information.

This, you're probably thinking, is precisely the kind of thinking that produced the classic Norton Utilities.TM

And you're right.

But even Peter Norton can't think of everything.

Which is why there's a built-in compiler for

creating databases of your own.

And why all Guides databases are compatible with the Instant Access program in your original package.

So you can add more languages without spending a lot more money.

To get more information, call your dealer. Or call Peter Norton at 213-453-2361.

And ask for some guidance.

Peter Norton

COMPUTING

dealers, or direct from Peter Norton Computing, Inc., 2210 Wilshire Blvd., #186, Santa Monica, CA 90403. 213-453-2361, Fax 213-453-6398, MCI Mail: PNCI © 1987 Peter Norton Computing

CIRCLE 103 ON READER SERVICE CARD

EDITORIAL

DOS Ex Machina

Any month now, we'll be greeted with the official debut of Microsoft's OS/2. It is, if you'll believe Bill Gates' patter, the operating system we've all been waiting for. At long last, we'll have multitasking and large program support. So why aren't we all more enthusiastic about it? Why is it that, as we get closer and closer to actually having OS/2, the alternatives begin to look so much better?

Some of this may be the result of the prolonged vaporware period. And some of it is no doubt the skepticism we normally feel at the introduction of any new software. In the case of OS/2, I think it's fair to say that the critics have had plenty to work with. There are all those minor technical points you've heard before: it's not thoroughly debugged yet; it's too fat; it's too slow; it's too expensive. Well, yes, true enough. But I'd like to remind everyone of a problem that dwarfs all those above.

Simply put, OS/2 is too late.

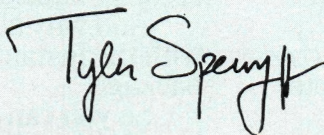
Note that I'm not talking about delays in shipping the product. Microsoft has been pretty good on meeting the ship dates for its OS/2 Developer Kits. No, what I'm talking about here is a critical marketing problem for OS/2: it's a dynamite product for a market that existed a couple of years ago.

The problem, as you've no doubt noticed, is that the world just didn't stop moving when the 80286 came along. As I write this, just weeks before Fall Comdex, there are several companies offering 80386 motherboard upgrades for XT owners. Intel is scheduled to announce their new Inboard 386/PC designed to fit into a normal PC. No toggle switches, no software tweaking. You just drop it in and your PC is an order of magnitude faster. (The RAM on the 386 card is well-used: some

for faster ROM BIOS access, and some for caching the hard disk.) And this for a price cheaper than you're used to seeing AT-clones sell for.

So, given that 386 revolution is well under way, you have to wonder about the leadership demonstrated in Microsoft's introducing a 286-bound operating system when everyone is gearing up for the 386 revolution. Probably the most telling indictment of Microsoft's strategy is the market's reaction to the introduction of its own Windows/386. Forecasting the success and failure of software is always a tricky matter, but it seems reasonable to presume that Windows/386 will be vastly more successful than OS/2 over the short run. Indeed, the release of Windows/386 before OS/2 with its Presentation Manager could be seen as Microsoft's tacit admission that an OS/2 tomorrow is no match for a DesqView today.

I mention all this because in the months to come, DDJ will be spending a great deal of space exploring the 386 universe. We'll be investigating a variety of operating systems and environments. And we'll be covering OS/2, of course. It's just sad to realize that our OS/2 coverage, even as it begins, may be as dated in another year as CPM Plus coverage is today.



Tyler Sperry
editor

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Editorial

Editor-in-Chief Michael Swaine
Editor Tyler Sperry
Managing Editor Vince Leone
Associate Editor Ron Copeland
Assistant Editor Sara Noah Ruddy
Technical Editors Allen Holub
Richard Relph
Contributing Editors Kent Porter
Namir Shamma
Ernest R. Tello
Rhoda Simmons

Copy Editor

Production

Director Art/Production Larry L. Clay
Art Director Michael Hollister
Assoc. Art Director Joe Sikoryak
Technical Illustrator Barbara Mautz
Typesetter Mary E. Lopez
Cover Photographer Michael Carr

Circulation

Circulation Director Maureen Kaminski
Fulfillment Coordinator Francesca Martin
Book Marketing Mgr. Jane Sharninghouse
Subscription Supervisor Kathleen Shay

Newsstand Sales

Coordinator Larry Hupman

Administration

Finance Director Kate Wheat
Business Manager Betty Trickett
Accounts Payable Supv. Mayda Lopez-Quintana
Accts. Receivable Supv. Laura DiLazzaro

Advertising Director

Ferris Ferdon (415) 366-3600
Marketing Mgr. Michael Wiener
Trafficking Coordinator Patricia Albert
Account Managers see page 129

Associate Publisher

Michael Swaine
Assistant Sara Noah Ruddy

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. DDJ is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

Article Submissions: Send manuscripts and disk (with article and listings) to the Associate Editor.

DDJ on CompuServe: Type GO DDJ

Address Correction Request: Postmaster: Send Form 3579 to Dr. Dobb's Journal, P.O. Box 27809, San Diego, CA 92128.

ISSN 088-3076

Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

Subscriptions: \$29.97 per 1 year; \$56.97 for 2 years. Canada and Mexico add \$27 per year airmail or \$10 per year surface. All other countries add \$27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.



M&T Publishing Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President and Publisher Laird Fosha

E=M C AZTEC

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple II C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

FREE 2-DAY DELIVERY!

Aztec C86 4.1 New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System ... \$199

- optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System \$299

- includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System. \$499

- includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: • Essential Graphics • C Utility Library • Curses • Greenleaf Communication, General, and Data Window • Halo • Panel+ • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • db-Query • Phact • Plink-86 Plus • c-tree • r-tree • Pmate

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM) \$349

Aztec C II-d (CP/M-80) \$199

Aztec C80 (TRS-80 3&4) \$199

Aztec C68k/Am 3.4 New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional \$199

A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer \$299

The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial \$499

Aztec C68k/Am-d plus source for the libraries and one year of updates.

FREE 2-DAY DELIVERY!

Aztec C68k/Mac 3.4 New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional \$199

- optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer \$299

The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial \$499

Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65 New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial \$299

- runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer \$199

- runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under \$1,000 outperform systems priced at over \$10,000.

Initial Host Plus Target \$750

Additional Targets \$500

ROM Support Package \$500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime PC/MS-DOS • Macintosh Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as \$500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime \$75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus \$10. Site licenses, OEM, educational, and multiple copy discounts are available.

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MANX

MS is a registered TM of Microsoft, Inc. • CP/M TM DRI • HALO TM Media Cybernetics • PANEL TM Roundhill Computer Systems, Ltd. • PHACT TM PHACT Assoc. • PRE-C, Plink-86, Plink-86+ • P-Force TM Phoenix • db Vista TM Ramna Corp. • C-terp, PC-lint, TM Gimpel Software • C-tree TM Faircom, Inc. • Windows for C, Windows for DATA TM Creative Solutions • Apple II, Macintosh TM Apple, Inc. • TRS-80 TM Radio Shack • Amiga TM Commodore Int'l. • Unix TM AT&T • Vax TM DEC • Aztec TM Manx Software Systems.

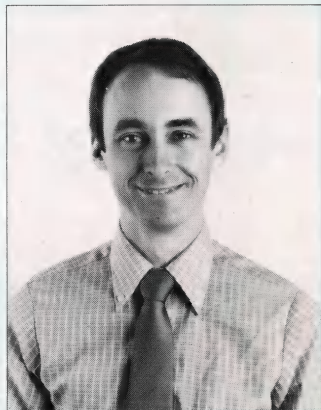
CIRCLE 104 ON READER SERVICE CARD

Manx Software Systems

1 Industrial Way, Eatontown, NJ 07724

It's been a crazy month, no doubt about it. I'd tell you all about it, except you'd probably be bored with the details, and as the old joke goes, I don't want to talk about it. Suffice to say that this month has seen the departure of Vince Leone, our long-suffering Managing Editor, and the usual mad rush to get the magazine out the door has been unusually mad.

This month has also seen an abnormal mix of last minute code changes and too little space in the magazine. The code for Rick Naro's LOCATE utility, for example, was updated for MASM version 5.0, but unfortunately we've had to continue the listings into next month. That



code will be joined by some last-minute additions to the articles by Dave Cortesi and Martin Tracy. Those of you who use CompuServe won't be affected by most of this madness since we'll be updating the code before it goes online. And before it goes

onto the listings disk.

One last note before I go: the gremlins obliterated my CompuServe number last month, but you can still reach me as 76703,4266. If you're doing neat things in Object-oriented programming or AI, now's the time to pitch me an article.

Tyler Sperry

Tyler Sperry
editor

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

(Act of August 12, 1970, Section 3685, Title 39, United States Code)

1. Title of Publication: Dr. Dobb's Journal of Software Tools, Publication No. 08883076.
2. Date of Filing: October 10, 1987.
3. Frequency of Issue: Monthly (12 issues, \$29.97).
4. Location of Known Office of Publication: 501 Galveston Dr., Redwood City, CA 94063.
5. Location of Headquarters of General Business Offices of the Publishers: 501 Galveston Dr., Redwood City, CA 94063.
6. Names and Addresses of Publisher, Editor, and Managing Editor: Publisher, Laird Foshay, 501 Galveston Dr., Redwood City, CA 94063. Editor, Michael Swaine, 501 Galveston Dr., Redwood City, CA 94063. Managing Editor, Vince Leone, 501 Galveston Dr., Redwood City, CA 94063.
7. Owner: M&T Publishing, 501 Galveston Dr., Redwood City, CA 94063.
8. Known Bondholders, Mortgages, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages or Other Securities: Markt & Technik, Hans-Pinsel-Strasse 2, 8013 Haar bei Munich W. Germany.
9. Extent and Nature of Circulation:
 - A. Total Number of copies printed. Average number of copies each issue during the preceding 12 months: 70,701. Actual number of copies of single issue published nearest to filing date: 76,069.
 - B. Paid Circulation. 1. Sales through dealers and carriers, street vendors, and counter sales. Average number of copies each issue during preceding 12 months: 14,727. Actual number of copies of single issue published nearest to filing date: 17,258. 2. Mail subscriptions. Average number of copies each issue during pre-

- ceding 12 months: 41,486. Actual number of copies of single issue published nearest to filing date: 42,697.
- C. Total Paid Circulation. Average number of copies each issue during preceding 12 months: 56,213. Actual number of copies of single issue published nearest to filing date: 59,955.
- D. Free distribution by mail, carrier, or other means, samples, complimentary, and other free copies. Average number of copies each issue during preceding 12 months: 2,266. Actual number of copies of single issue published nearest to filing date: 1,900.
- E. Total distribution. Average number of copies each issue during preceding 12 months: 58,479. Actual number of copies of single issue published nearest to the filing date: 61,855.
- F. Copies not distributed. 1. Office use, left over, unaccounted, spoiled after printing. Average number of copies each issue during preceding 12 months: 650. Actual number of copies of single issue published nearest to filing date: 655. Returns from News Agents. Average number of copies each issue during preceding 12 months: 11,572. Actual number of copies of single issue published nearest to filing date: 13,559.
- G. Total. Average number of copies each issue during preceding 12 months: 70,701. Actual number of copies of single issue published nearest to filing date: 76,069.

I certify that the statements made by me above are correct and complete.

Laird Foshay, Publisher

Ten Years Ago in DDJ

"DALLAS, Texas—The first flexible disk drive for 5¼" diskettes to offer double density recording of 250,000 bytes on each side of a diskette, was introduced at the National Computer Conference here today by the Pertec Division of Pertec Computer Corporation." *News release received Jun 18, 1977, DDJ, November/December 1977.*

Kicking the 8080 Habit

"Maybe it's just my imagination, but it seems that a lot of people aren't utilizing the Z-80 to its fullest. Everyone is so used to writing code for the 8080 that they don't seem to bother upgrading their software when they upgrade their CPU..."

"I would like to see you guys... explain all the nifty Z-80 tricks. I know I can't be the only one that is stuck in the rut of 8080 code. (Please!! Don't tell me I swapped my CPU board JUST for speed—the software potential is fantastic.)" *Letters to the Editor, DDJ, November/December 1977.*

Exec with Extreme Prejudice

"In a multi-tasking environment such as we expect to see in MS-DOS 3.0, this function (function 4BH - EXEC) will be even more useful and will undoubtedly be elaborated with several additional features. Under such an operating system, a parent task can 'spawn' any number of child tasks, which can execute concurrently and asynchronously, and communicate by means of queues, semaphores, and pipes.

"Well, you say, pie in the sky is all very nice, but why is the EXEC function getting so much attention in this magazine column? The answer is, of course, that when I tried to actually use the function I ran into any number of glitches and hazy spots in the documentation." *Ray Duncan, "16-bit Software Toolbox," DDJ, December 1983.*

Encryption

"A small but vital piece of hardware containing a microelectronic chip only 1 cm square has been tested and validated at the Commerce Department's National Bureau of Standards (NBS)—marking the first NBS validation of a commercial implementation of the Federal Data Encryption Standard published early this year." *News release received October 31, 1977, DDJ, November/December 1977.*

DR. DOBB'S JOURNAL of
COMPUTER
Calisthenics & Orthodontia

Running Light Without Overbyte

SOFTWARE ENGINEERING COMES OF AGE.

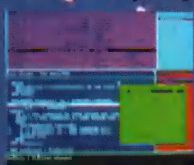
ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.

**FREE TURBO PASCAL
TO LOGITECH MODULA-2
TRANSLATOR**

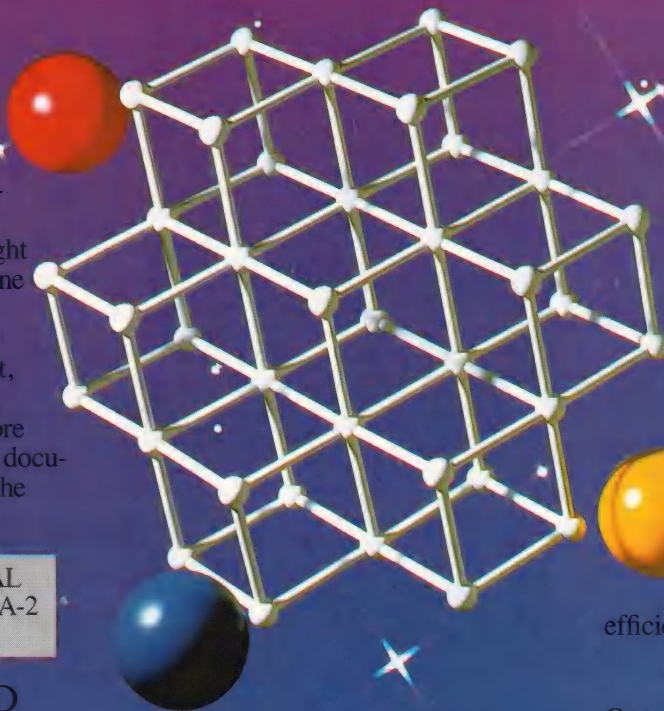
NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined breakpoints. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.



NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.



NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET of CHAR. Provides optimization for tighter, more efficient code generation.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:

800-231-7717

In California:

800-552-8885

- ☐ **LOGITECH Modula-2 V. 3.0 Compiler Pack** **\$99**
Compiler in overlay and fully linked form. Linkable Library, Post Mortem Debugger, Point Editor
- ☐ **LOGITECH Modula-2 V. 3.0 Toolkit** **\$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter
- ☐ **LOGITECH Modula-2 V. 3.0 Development System** **\$249**
Compiler Pack plus Toolkit
- ☐ **Turbo Pascal to Modula-2 Translator** **FREE**
With Compiler Pack or Development System
- ☐ **Window Package** **\$49**
Build true windowing into your Modula-2 code.
- ☐ **Upgrade Package**
Call LOGITECH for information or to receive an order form.

Add \$6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed \$

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

LOGITECH

LOGITECH, Inc.

6505 Kaiser Drive, Fremont, CA 94555

Tel: 415-795-8500

In Europe: LOGITECH, Switzerland

Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K.

Tel: 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equipment Corp.

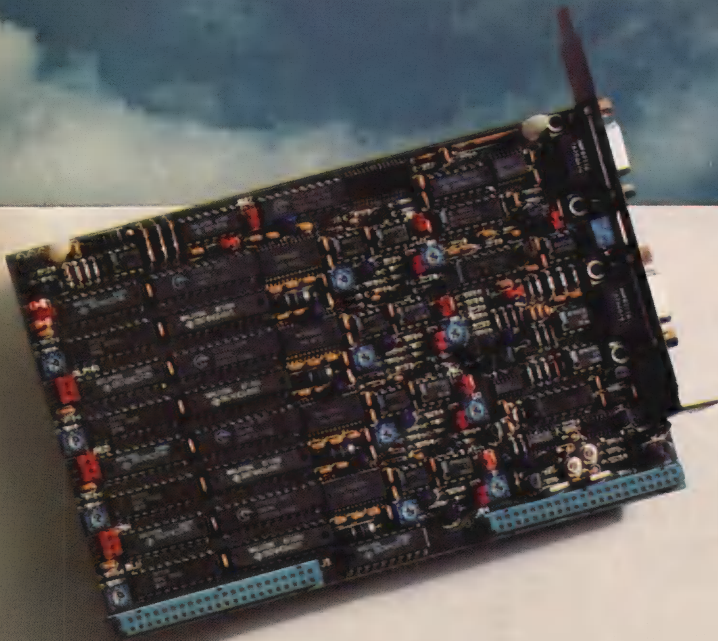
CIRCLE 105 ON READER SERVICE CARD

Now Taking Applications.

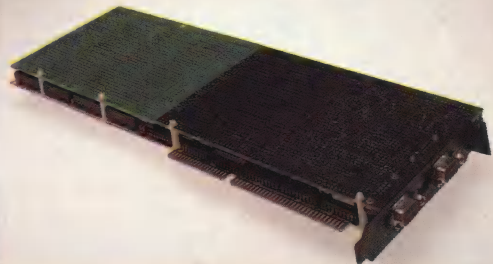


Take a look at the specs on VISTA™, a good look. Notice the processing, programming, and video capabilities? Now think real hard about what *you* could do with the power of VISTA and a microcomputer. Incorporate it with your system to create a digital pre-press proofing station for publishing. Design a graphics workstation which outputs both colorful hi-resolution slides and broadcast-quality animated images. Construct a CAD system which merges computer generated images with real-life backdrops for architecture, packaging or other industries. And, after you've brainstormed your way to new horizons of videographics possibilities, get your own VISTA and start working.

Introducing VISTA™ Videographics.



VISTA is a powerful single-slot videographics adapter which captures and displays video signals in real time.



Let's Get Specific.

We knew you couldn't resist seeing the facts, and frankly, our engineers wouldn't have it any other way. Here is an overview of VISTA's key features.

FEATURES:

- 4Mbytes of Video RAM on-board
- Texas Instruments' TMS 34010 GSP
- Flexible, programmable resolutions
- NTSC and PAL compatible
- Four 8-bit channels for real-time capture
- Fully integrated genlock
- Processor memory expandable in 2Mbyte increments to 12Mbytes
- Four 2K x 8-bit CMOS static RAM LUTs
- Display can be color-mapped, RGB, or a versatile combination of both
- Interlaced and non-interlaced display
- Binary and fractional programmable zoom capability, creates horizontal and vertical magnify or minify
- Smooth horizontal and vertical programmable panning, includes wrap-around and split screen
- Suggested Retail Price: \$5995.

ADDRESSABLE RESOLUTIONS:

32 bits/pixel	16 bits/pixel	8 bits/pixel
1024x1024	2048x1024	4096x1024
512x2048	1024x2048	2048x2048
256x4096	512x4096	1024x4096

CAPTURE RESOLUTIONS:*

NTSC	PAL
(RS-170A)	(CCIR-624)
756x486	738x576
604x486	590x576
504x486	492x576
432x486	422x576

*Resolutions are programmable; these are nominal ones for interlaced NTSC and PAL compatible.

DISPLAY RESOLUTIONS:*

NTSC	PAL	Interlaced	Non-Interlaced
(RS-170A)	(CCIR-624)		
1512x486	1476x576	1024x768	768x576
1008x486	984x576	(60 Hz)	(50 Hz)
756x486	738x576		
604x486	590x576	768x768	756x486
504x486	492x576	(80 Hz)	(60 Hz)

*Resolutions are programmable; these are nominal ones.

COMPUTER REQUIREMENTS:

Host Type:	IBM PC AT and 100% Compatibles, Compaq 386, Apollo DN 3000-single-slot board
Data Bus:	16-bit or 8-bit (self-configuring)
Bus Clock:	6MHz to 12MHz
Power Consumption:	15 Watts

It's So Flexible, We've Added Support.

With its Texas Instruments TMS 34010 graphics processor, large quantity of video memory, and proprietary video cross-point, VISTA can be programmed for an array of powerful market-specific videographic applications. To help you maximize VISTA's potential, Truevision offers a range of C-language programming tools for developers. And when your system is market-ready, we'll support your marketing efforts with our **TRUEVISION SOFTWARE CATALOG**, **TRUEVISION NEWS**, and **THE PULSE**.

We're For Higher

Resolution... Power... Flexibility... Quality. Join the many key manufacturers and developers already working with the state of the videographics art, VISTA. Call us at **800/858-TRUE** for more information on the VISTA Developer's Program. We're ready to take your application today.

AT&T
Electronic Photography and Imaging Center
7351 Shadeland Station, Suite 100
Indianapolis, IN 46256
800/858-TRUE



CIRCLE 106 ON READER SERVICE CARD

IBM is a registered trademark of International Business Machines Corp. **International Inquiries:** contact Techexport at 617/890-6507 (USA), or London at 44-1-991-0121. In Italy, contact S/RIO Informatica at 39-2-301-0051. Suggested retail price is US domestic price.

LETTERS

**Stone Age Software**

Dear DDJ,

Thank you for an excellent editorial in the September 1987 issue. The point Tyler Sperry makes about the crippling of potentially powerful machines is a sore point with me also.

I work for a large computer manufacturer and have seen for myself how these limiting factors get incorporated into machines, not only in software but also in hardware. I don't see an immediate solution but, like you, I'll keep screaming for the liberation of these systems from stupid design flaws.

Les J. Record

1201 East Mesa Pk. Dr.
Round Rock, TX 78664

Dear DDJ,

I don't think there is any reason to complain about the architecture of the 80286, at least not about the inability to switch back to real mode. This is like building a V30-based computer to run CP/M Plus and then cursing at those crazy 16-bit data and I/O buses that make designing a good 8-bit system such a mess.

Remember, the 8086/8088 didn't even have an 8080 emulation mode, so why blame the designers of the 80286 for including an 8086 emulation mode? It is there just to give an upward migration path, not to be frantically switched on and off.

Let's face it, it's not the chip that is Stone Age but the

software. Switching back from protected mode itself is Stone Age, not the messy way it has to be done. And AboveBoard emulators are a Stone Age way of reducing a 16-megabyte address space to an 8-megabyte collection of memory chunks—and there are people proud of that!

The cry for hardware compatibility—which means hardware-dependent software (which equals incompatible software)—has made the transition from today's status quo far more difficult than the change from 8-bit CPM to 16-bit MS-DOS has been. It's like using metal tools to make better stone axes. It will take true Bronze Age men to change things.

Jost Riedel

Am Reservoir 2

P.O. Box 1141

D-3522 Bad Karlshafen 1

West Germany

Recursive/Iterative Trade-Off

Dear DDJ,

I really enjoyed the article on backtracking by Charles F. Bowman (August 1987). His structure for the

puzzle-solving program, using recursive techniques, is nice and simple.

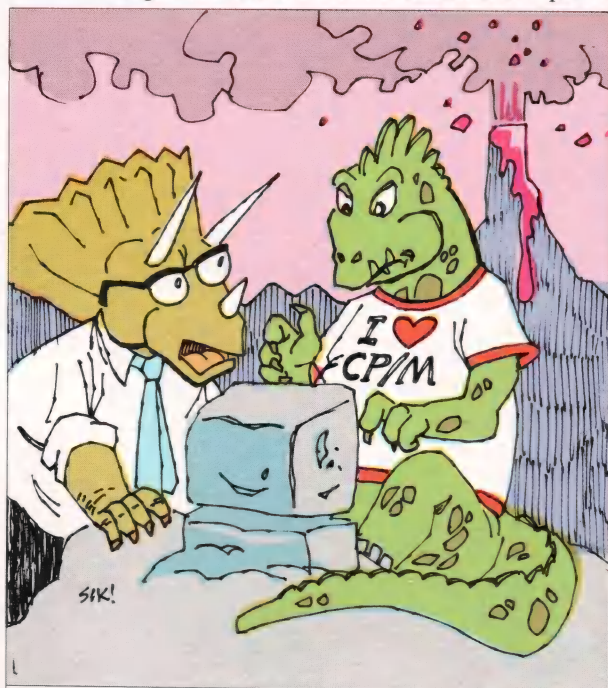
I just have one quibble with the article: In his discussion on ways to speed up the process, Bowman apologizes for his use of a recursive approach and states the conventional wisdom, "Recursive procedures are costly because of the considerable amount of overhead required for each successive call. Your program must save registers, store a return address, allocate local storage, and so on."

That statement surely sounds plausible. I've accepted it on faith for years. It was probably even true on most of the old mainframe computers, which is probably the context in which both Bowman and I heard it. But it may not be true in the context of microcomputers. All current microprocessors support very fast subroutine call/return mechanisms as well as stack push/pops. Because of these operations, sometimes recursive is better in all ways.

To test the truth of the recursive/iterative trade-off, I wrote an eight-queens problem in both forms (the recursive version is shown in Example 1, page 14). The program was written in Turbo Pascal for a PC clone. Much to my pleasant surprise, the recursive version turned out to be a full 40 percent faster than the non-recursive form. It was also smaller, of course.

Another concern often voiced about recursive approaches is that of limited stack space. The idea is that, if your program has to go many levels of recursion, it may crash by overflowing the stack. To test that hypothesis, I ran the program shown in Example 2, page 14, again using Turbo Pascal. The program did indeed crash (gracefully) but at a level of more than 5,400 layers of recursion. That's 5,400 successive subroutine calls, folks! That should be enough for most of us!

So it appears that the conventional wisdom "recursion



Hurry up Jennings—our very existence may depend on it!

How to tell the difference between DESQview™ 2.0 and any other environment.

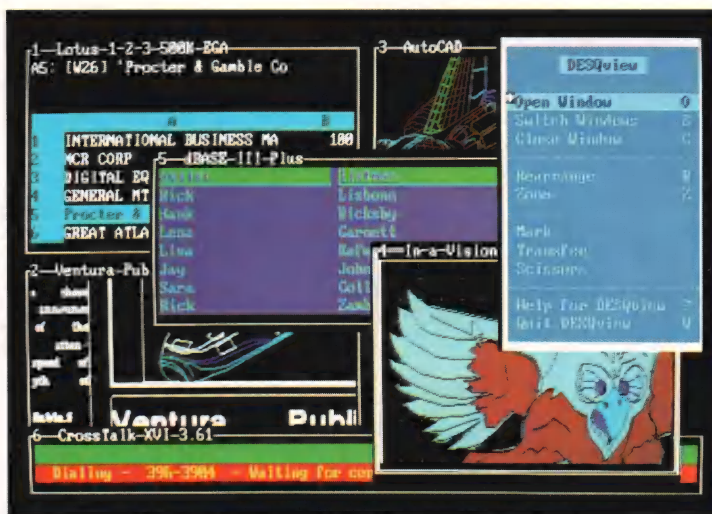
Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM™, Topview™, and Microsoft Windows™ specific programs. In some cases you'll add as little as 10-40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"I wouldn't want to run an IBM



One picture is worth a thousand promises.

or compatible computer without DESQview"—InfoWorld, Michael Miller.

"A colossus among windowing environments"... "will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICRO-TIMES, Birell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS

IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5 1/4" or 3 1/2" floppy diskettes

Rush me DESQview 2.0! Today!				DDJ12/87
No. of Copies	Media 3 1/2" 5 1/4"	Product	Retail Price ea.	Total
		DESQview 2.0	\$129.95	\$
Shipping & Handling			USA \$ 5.00	\$
			Outside USA \$ 10.00	\$
Sales Tax (CA residents)			6.5%	\$
Payment: <input type="checkbox"/> Visa <input type="checkbox"/> MC <input type="checkbox"/> AMEX <input type="checkbox"/> Check			Amount Enclosed	\$
Credit Card: Valid Since _____ / _____ Expiration _____ / _____				
Card Number: _____				
Credit Card Name _____				
Shipping Address: _____				
City _____ State _____ Zip _____ Telephone _____				
Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.				
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.				



Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation. Hayes is a trademark of Hayes MicroComputer Products Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.


```

{ A skeleton program for the eight-queens
  problem. By altering the dimensions and the
  routines fit, place, and unplace, it also
  serves as a model for any other solution
  involving backtracking. The program assumes a
  data array q[0..maxcount] containing the
  values to be adjusted (row on the chessboard
  in the case of the eight queens. Three
  procedures are assumed:
  Function fit(1) compares the position of the
  queen q[1] with all pieces placed so far,
  and returns TRUE if there is no conflict.
  Procedure place(1) records queen q[1] on the
  board
  Procedure unplace(1) removes queen q[1].
}

procedure Try(1: integer);
begin
  for q[1] := 0 to maxcount do begin
    if fit(1) then begin
      place(1);
      if 1 = max then ShowResult
      else try(1 + 1);
      unplace(1);
    end;
  end;
end;

{ Main Program }

begin
  initialize;
  try(0);
end.

```

Example 1: Eight queens problem in Pascal

```

Program test;
procedure bump(n: integer);
begin
  writeln(n);
  bump(n + 1);
end

{ Main Program }

begin
  bump(0);
end.

```

Example 2: Program to test levels of procedure nesting

Bytes 8088 Clocks			
3	12+ea = 12+9 = 21	mov	ax, word ptr (bp).value
2	2	mov	bx, ax
3	12+ea = 12+9 = 21	mov	ax, word ptr (bp).value(2)
<hr/>			
8	44		
3	12+ea = 12+9 = 21	mov	ax, word ptr (bp).value(2)
3	12+ea = 12+9 = 21	mov	bx, word ptr (bp).value
<hr/>			
6	42		
3	24+ea = 24+9 = 33	les	bx, (bp).value
2	2	mov	ax, es
<hr/>			
5	35		

Table 1: Timings for Example 6, page 26, July 1987 DDJ

is costly" needs to be put to rest. In the modern world of micros, the simplest and most elegant solution may also be the smallest and fastest.

As an aside, the issue of fast call/return mechanisms should also cause us to take another hard look at assembly-language programming. The conventional picture of an assembly-language program is that of one long string of in-line code and macros. Many programmers tend to think this is necessary to achieve the speed you expect of a native-language program. But because of the support provided by the micro chips, the trade-offs of modularity vs. speed favor the former in assembly language, even more than in a higher-order language.

DDJ Forum User

Instruction Timings

Dear DDJ,

A reader called to point out an error in my article "8088 Assembly-Language Programming Techniques" (July 1987). It seems that the early versions of Intel's instruction timing tables implied that no cycles are used for effective address calculation when the AX register is used for moves to and from memory. In fact, this is the case only when using direct memory reference. For all other cases, AX is treated in the same way as any other register.

Table 1, left, gives the timings for my Example 6, on page 26 of the July issue. There is no doubt that the third method is best for real mode. Protected mode on the 80286, however, is another story.

Tom Disque
SAS Institute Inc.
P.O. Box 8000
SAS Circle
Cary, NC 27511-8000

DDJ

The Leaders Made PVCS The Leading Source Code Control System.

NOW
VAX/VMS
& MS-DOS versions

When it comes to maintaining their most valuable asset, the leading software publishers rely on the POLYTRON Version Control System (PVCS). From accounting firms to airlines, the leading service companies depend on PVCS to maintain the integrity of their programs. Leading manufacturing companies use PVCS to maintain their state-of-the-art software. Leading high technology companies turn to PVCS to handle configuration management for software projects that represent an investment of hundreds of thousands of dollars. The largest aerospace companies and defense contractors use PVCS to maintain integrity of projects during development and after delivery of software. Independent programmers use PVCS to improve their productivity and software quality for themselves and their clients.

Simplify Configuration Management

When large and complex software programs are being developed on personal computers or VAX minicomputers, effective management of the revisions and versions becomes critical. PVCS simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of programmers and managers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

- Storage & Retrieval of multiple revisions of text.
- Maintenance of a complete history of changes.
- Maintenance of separate lines of development using branching.
- Merging simultaneous changes.
- Resolution of Access Conflicts.
- Modules can be retrieved by their own revision number, system version name, or specified date.
- Uses "reverse deltas" to rebuild a prior version making PVCS the fastest version control system over the project life cycle.
- Projects already under development or in the maintenance stage can be easily put under the control of PVCS.

Manages Development On Local Area Networks

Programming teams using Local Area Networks depend on PVCS to help the managers and team members work together. In fact, Novell and 3Com themselves depend on PVCS to manage the versions of their own network software products.

Supports MS-DOS and VAX/VMS Development

Now, companies that develop software on VAX systems running VMS can also use PVCS. And since the VMS and MS-DOS versions of PVCS use the same "logfile" format, you can easily develop software on PCs and maintain the code on the VAX or vice versa. The menu-driven, screen-oriented interface (and optional command-driven interface) makes it easy for programmers and librarians or administrators to use PVCS on a PC or VAX or both systems.

PVCS Maintains System Integrity

PVCS prevents corruption of code that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

PVCS & PolyMake Work Together

PolyMake, the leading MS-DOS make utility, is now available for the VMS operating system. This allows you to write makefiles that will function in both PC and VAX environments. Additionally, PolyMake reads time & date stamps of PVCS archives for fast, accurate program rebuilding.

PVCS and PolyMake Maintain Source Code Written In Any Language.

Only PVCS meets the needs of independent programmers and corporations. Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced MS-DOS version of PVCS.

Personal PVCS — Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths (e.g. new versions for different host systems, or a new program based on an existing program).

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

PVCS for VAX systems — Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

The Preferred Version Control System

The customers listed below are just a few of the innovative leaders that have made PVCS the leading version control program for personal computers.

Alcoa Aluminum
Arthur Anderson
AT&T
Ashton-Tate
Bank of America
Bell Labs
Bendix
Boeing
CIGNA
Citibank
3Com
Colonial Penn
Commerce Clearing House
Control Data Corp.
Corvus
CXI
Digital Equipment Corp.
Deloitte Haskins + Sells
Diebold
Dow
Dunn & Bradstreet
EDS
Educational Testing Service
E-Systems
Equitable Life
Federal Express
First Boston
Ford
Fox Software
Fujitsu
GTE
Haelees
Hewlett-Packard
Honeywell
Hughes Aircraft
IBM
Industrial Networking
Intel

ISC Aerospace
IVAC
Javelin
Lattice
Lawrence Livermore
Lotus
McData Corp.
McDonnell Douglas
Mead Data Central
MIT Lincoln Labs
Nastec
Novell
NCR Technologies
Pitney Bowes
Plexus Computers
Price Waterhouse
ROLM
Rockwell International
Safeco
Sears
Security Pacific
Sperry
Software Publishing
Spacelabs
Standard Oil
Standard & Poors
Tandem
Tektronix
Telex
Texas Instruments
Touche Ross
Unisys
United Airlines
United Parcel Service
United Technologies
U.S. West
Westinghouse Electronics
Xerox

	MS-DOS*	VMS		
	PC/XT/AT	Micro VAX II	VAX 7xx	VAX 8xxx
Personal PVCS	\$149			
Corporate PVCS	\$395			
Network PVCS	\$995**	\$4,950	\$9,500	\$10,500+
PolyMake	\$149			
Network PolyMake	\$447**	\$1,250	\$2,375	\$2,500+

*Compatible with MS-DOS 2.0 through 3.3.
Compatible with the IBM PC/XT/AT & other MS-DOS PCs.

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
VISA/MC 1-800-547-4000.

Dept. No. 355
Oregon & Outside USA call (503) 645-1150.
Send Checks, P.O.s to: POLYTRON
Corporation, 1815 NW 169th Place,
Suite 2110, Beaverton, OR 97006.

POLYTRON

High Quality Software Since 1982

CIRCLE 108 ON READER SERVICE CARD

Texas Instruments has system developers need.



“Personal Consultant™ Plus... offers a very fine expert system development and delivery tool that already has a proven record with end-users.”

— Susan Shepard, *AI Expert*

Personal Consultant Plus 3.0 Standard Features

- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

Personal Consultant Images

- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

what serious expert Power tools.

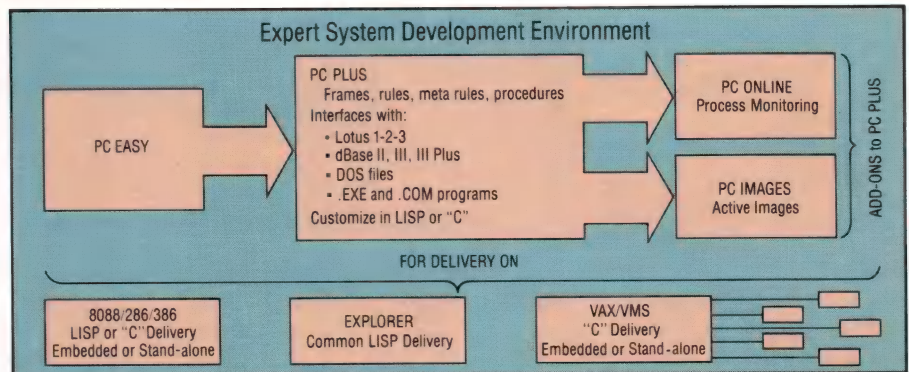
Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.

package helps deliver expertise that is "online all the time."

Application delivery as flexible as the tools themselves.

Delivery can be in LISP for flexibility, or "C" for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.



Personal Consultant Plus. Full power for an affordable price.

At \$2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.

At \$495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process.

At \$995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."

— Jim Seymour, *PC Magazine*.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

Pick up the phone and gain a powerful advantage.

Call 1-800-527-3500 for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106

© 1987 TI

Personal Consultant and Explorer are trademarks of

Texas Instruments Incorporated.

dBase is a trademark of Ashton-Tate.

Lotus 1-2-3 is a trademark of Lotus Development Corp.

VAX and VMS are trademarks of Digital Equipment Corporation.

*Available 4Q 1987.

- knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online

- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

**TEXAS
INSTRUMENTS**

Dynamic Linking in OS/2

by David E. Cortesi

No operating system is likely to include all the functions that creative programmers will demand. It appears that Microsoft, benefiting from past experience, has purposely designed OS/2 so that third-party developers can more easily add functions to it. Yet to maintain an acceptable degree of quality assurance, Microsoft wanted to make sure that any additions would fit seamlessly into the whole system without compromising its reliability. This article is a brief survey of the mechanisms included in OS/2 to allow this, and in particular, the concept of dynamic linking.

Briefly put, a dynamic link is an external reference that is not resolved at the time the program is linked. Instead, the connection to the external routine is made either while the program is being loaded or sometimes even later while it is executing.

Dynamic linking isn't a novel idea; it was fundamental to the operation of the Multics system (the Intel 80286 has some intriguing similarities to the GE 645, the Multics host) and readers of the DDJ Forum on CompuServe have told me that the Prime operating system PRIMOS and UCSD Pascal have similar features.

Preparing a Dynalink Library

Here's how dynamic linking works in OS/2. You design and build a package of code that will be useful to more than one program. At first you store its object code in an object library as usual and test it by linking it in the usual way with the programs that use it. When your package is in something like its final form, you run it

Built-in facilities for the third-party extension of OS/2

alone through the linker in a special pass. You also supply a description file that tells the linker that this code will be a dynamic link library, or a dynalink, as it's come to be called. In the description file you specify the names of all the entry points that will be publicly available in this package. You may also specify attributes for individual segments; I'll come back to those later.

The linker processes the object code much as it does in MS-DOS, merging segments by class and group, resolving internal references between segments, adjusting offsets to account for merged segments. One step of linking in OS/2 is different from that with MS-DOS. In MS-DOS, the linker's output is a single, monolithic binary image in which the input segments have lost their individual identity. In OS/2, the linker keeps segments separate in the executable file. That's necessary so that the system loader can load a program one segment at a time, as it must do in order to build a local descriptor table (LDT) for the 80286 hardware.

The processed segments of your package of subroutines go into a file that has the same format as an OS/2 executable (.EXE) file, but by convention a dynalink library has the file type description. If your package is THEGOODS, its code will be linked as THEGOODS.DLL. That is all that needs to be done for code that will be bound to its caller very late, during execution time, but that's a rare use.

Most dynalink libraries need to be known to their client programs at link time. In that case, one more processing step has to be done. A utility called IMPLIB reads the description file and writes an artificial object library that can be used at link time. Applied to the

David E. Cortesi, 415 Cambridge St., #18, Palo Alto, CA 94306. Dave is a former DDJ columnist.



description file for your package, this produces a very small file named THEGOODS.LIB.

Using a Dynalink Library

Now client programs can begin using your package. They declare its entry points just as they would declare any other external references, whether in assembly language:

extern GoodyA.far

or in Pascal:

```
Procedure GoodyB(c:char);  
  external;
```

or in C:

```
extern int far GoodyC( );
```

And the calls to the package's procedures are written just as they would be if the package were to be linked in the usual way (which it might still be in some cases).

When a client program is linked, the import library, THEGOODS.LIB, is one of the libraries input to the link. The object records in it are special. They don't contain object code; they only tell the linker the name of the dynalink library (THEGOODS.DLL, remember?) and the names of the entry points that it exports for use. The linker writes a table of these names into the linked program (let's say it's CLIENT.EXE) and pointers to where the references occur in the linked segments.

Load Time Linking

Eventually CLIENT.EXE will be loaded for execution. The linked segments of code and data will be brought in from the disk file—if necessary. It might not be neces-

sary because it is possible to tell the linker to mark any segment "load on demand." In that case, the OS/2 loader won't load the segment but will only set up the local descriptor table to cause a hardware trap if the segment is referenced by an instruction. When and if that happens, the segment will be loaded.

Once it has storage copies of the program's segments, the loader processes dynamic links. The linker has given it the character string file name of the dynalink library and the names of the entry points needed. The loader looks up the file; it has to be found in a directory designated in the system configuration file.

The system may not have to load the segments of dynalink code. Dynamically linked code segments are shared among all clients that use them, so if CLIENT.EXE is the second instance of a program that uses THEGOODS, no disk input will be needed for code segments.

Because a dynalink library has precisely the same file format as a .EXE file, the process of loading dynalink code is really just an extension of loading a program. The only extra step is that the loader has to fix up the external references in the client code. Because the process is so similar, there's no reason why code in one dynalink library shouldn't call code in another one, and that in a third, and so on.

Benefits of Dynamic Linking

Some of the benefits of dynamic linking are clear at once. There's an economy of disk space: whereas under MS-DOS every client program contains a copy of the common code, OS/2 stores only a single copy in the dynalink library. There's economy of memory because only a single copy of common code is kept in storage. There's economy of load time because only the needed segments are brought in from disk. (What's faster than a disk cache? Not doing the disk input at all!)

These features would be enough to justify extensive use of dynamic linking. And OS/2 does use it extensively. The whole interface to the operating system is based on it. All the OS/2 system functions are presented as external procedures that programs declare and call. And all of those system procedures are defined by code in dynalink libraries with names such as DOSCALLS.DLL, VIOCALLS.DLL, and MOUCALLS.DLL. As a result (and in sharp contrast to MS-DOS), it is actually easier to use OS/2 system calls from a high-level language than it is to use them from assembly language.

I promised this would be a discussion of system extensions; here's where the connection is made. Because the entire programming interface to OS/2 is through dynalinks—and because dynalinks may themselves call DynaLinks—any new dynalink library is a functional extension to OS/2 on an equal footing with the system code itself. There aren't any special interfaces, no magic incantations that only gurus may use; there isn't the sharp distinction between, say, ordinary programs and TSR programs that exists in MS-DOS. There is just the hierarchy of functions available in DynaLink libraries, with the OS/2 kernel dynalinks at the base of the pyramid.

And there is no lack of function, either. Don't suppose that, because a system extension is restricted to the same facilities that any program might use, it isn't possible to write interesting system extensions. There are a few limitations, and I'll mention them shortly. But consider this: all the functional extensions to be supplied in the IBM Extended Edition of OS/2 (database facility and multiple communications protocols) as well as the whole of the Microsoft Presentation Manager (protect-mode Windows), all of this code will be supplied as dynalink libraries. There won't be any features added to OS/2 at a kernel level to support them—they're all there in 1.0 and available (to high-level languages, remember) as dynalink calls. So if you repackage your B-tree access method as a dynalink library, it will use the same kernel functions and be exactly as accessible to its clients as IBM's database facility is to its.

Let's explore some of the subtleties of dynamic linking. Return to the point at which you are linking your package of code into a dynalink file. A "definition" file must be given to the linker to describe it. A definition file may be used when linking any program; that's how you tell the linker to mark a program segment for deferred loading. But there are two definitions that will most often be applied to DynaLink segments—one is shared data; the other is IOPL code.

Shared Data Segments

By default, data segments are not shared between programs. Instead, a new copy of a data segment is loaded from the .EXE or .DLL file for each instance of a program that uses it. That fits the expectations of most programs: you don't ordinarily think of a segment of data as being accessible to two or more concurrent programs at once.

You can tell the linker to mark a data segment

"single," however—that is, that there is to be only one instance of that data segment in storage no matter how many client programs might have concurrent access to it. This simple concept forms the basis for some sophisticated applications.

Recall that, although such a data segment is common to all client programs, it isn't directly accessible to them. The data segment is linked with your code, not (directly) with the client program. If it doesn't contain any exported entry points, there is no way in which a client program can form an external reference to it. Though there are ways in which determined programmers could find out its segment address, there is no way they could find out how you arrange and manage its contents unless you tell them. The only convenient way that client code can access a dynalink data segment (shared or not) is by calling the procedures in the dynalink package.

Because a shared data segment is dynamically linked, it will be loaded with the first client program to request the package. It will remain in storage so long as at least one client program does. (If it's important to keep your dynalink package in storage at all times, you can write a dummy client program and start it with a statement in the configuration file.) The segment is an island of static data that your code can address easily (because it was linked with your code) but that *only* your code knows how to address. You can use it for a shared buffer pool, or for queues of work (however your package defines "work"), or generally for any kind of pooled resource that your package can usefully manage on behalf of multiple concurrent client programs.

You can't predict how many clients may be executing in your package at any instant; there might be none or there might be dozens! But the OS/2 kernel has plenty of functions to help you manage shared data in a multiprogramming environment. There's a complete, and quite efficient, set of semaphore operators, so you can serialize access to the data. There's a generalized queueing facility so that multiple "writer" threads can queue data (using a variety of queue disciplines) for a single "reader" to process. There's a storage suballocation facility that has built-in serialization so multiple threads can allocate and free pooled storage concurrently.

I/O Privilege and Devices

In OS/2, application code runs at privilege level 3, in the 80286 scheme of things. The I/O privilege level is set at 2, so application code that tries to do an I/O instruction will trap out and be terminated. Any code segment, however, dynalink or not, can be marked at link time as being eligible for I/O privilege. Code in such segments may request access to a range of I/O port numbers from the kernel and then may do I/O instructions, including setting and clearing the interrupt flag.

There are serious restrictions on this I/O privilege, however. There is no way that application code can get control on an I/O interrupt. There is no way that application code can lock a segment in storage so as to use it for a DMA buffer. And because OS/2 is slicing time among potentially many programs, polling an I/O port is



The New Standard Bearer.



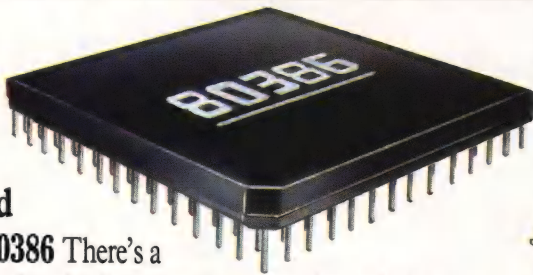
THE SOFTWARE LINK

CIRCLE 110 ON READER SERVICE CARD

A Number of Reasons A Number

1. Designed

for the 80386 There's a revolution taking place in desktop computing. A revolution that's been launched by a square wafer of silicon known as the 80386 microprocessor chip. It puts minicomputer potential at PC users' fingertips. It's a fact that virtually every leading PC manufacturer has built a "box" around this chip. And it's a fact that the "New Operating System" will, supposedly, even run on it. But, it's also a fact that *their* system wasn't designed for the 80386. Ours is. And it's called PC-MOS/386™



2. PC and PS/2 Compatible

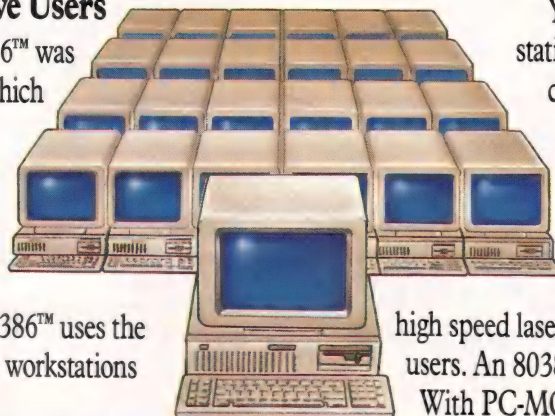
In designing PC-MOS, we knew our first priority was to exploit the minicomputer capabilities of 80386-based PCs & PS/2s. But we went further, and developed a system which would be fully compatible with the millions of existing PCs, PC ATs, and

nothing less from the new standard bearer.

3. One, Five, Up to Twenty-five Users

From the beginning, PC-MOS/386™ was designed as a versatile operating system which could support twenty-five users as easily as it supports one. The system comes in single, five, and 25-user modules, so you're able to start with what you need and expand when you're ready.

In a multi-user setting, PC-MOS/386™ uses the computing power of the host PC to drive workstations linked to standard RS-232 ports.



4. Thousands of DOS Programs PC-MOS/386™ gives you the best of the past, and the best for your future. Which means that while PC-MOS/386™ totally replaces your old DOS, you won't have to replace the programs you've spent a lot of time learning.

And it all happens so effortlessly. You'll continue to reap the benefits of your favorite DOS programs, while entering a new arena of power.

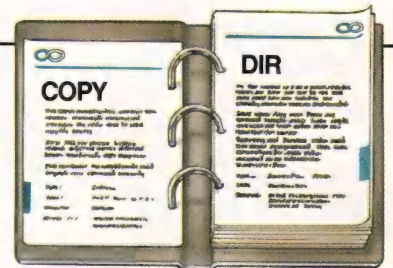
Think of it! Programs like dBASE III, WordPerfect, Lotus 1-2-3 and Symphony, WordStar, MultiMate...literally thousands of DOS programs—all compatible and multi-user available.



5. Familiar Commands Like DIR and COPY

Just as you don't have to learn a whole new array of software to take advantage of PC-MOS/386™, neither do you have to learn an entirely new set of commands.

Instead, the system builds on the knowledge you already have. "COPY" still copies files, and "DIR" still gives you a directory listing. As you might expect, we didn't stop there. There's a wealth of features that have strengthened the commands you know, making them more powerful and easier to use.

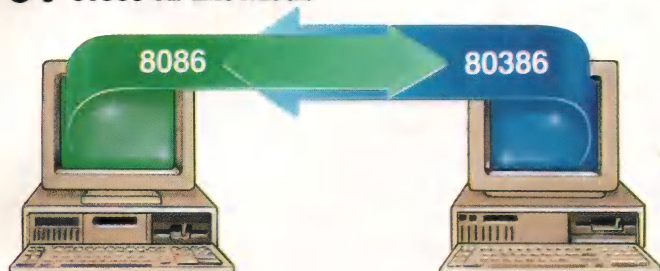


You can choose from a variety of workstations. Mix and match dumb terminals costing under \$500 each with PCs and PS/2s running our terminal emulation software.

All of the host's resources can be shared. Programs, data, hard disks, tape backup units & printers (including high speed laser printers) are suddenly available to all users. An 80386-PC has minicomputer potential. With PC-MOS/386™ you can "mini" your micro.

of Users Will Choose PC-MOS/386.™

6. Concurrently Supports Virtual 8086 and 80386 32-Bit Mode



80386-based PCs & PS/2s are dual-personality computers. To run DOS programs, they act as PCs with a 640K memory limit. But to take advantage of their minicomputer capacity, they operate in true 80386 mode which lets them address up to four gigabytes of memory. PC-MOS enables the 80386-host and its workstations to independently switch between these modes—making DOS compatibility and 80386 power simultaneously possible.

7. Multi-Tasking

While it's true you could look elsewhere for multi-tasking, why would you want to? The *other* multi-tasking operating system is not now, nor is it planned to be, multi-user. It won't even run multiple DOS applications in multi-tasking mode.

Now consider PC-MOS/386.™ At the touch of a key, you can switch between up to 25 different tasks. And if you have workstations connected to a host, they get multi-tasking, too. Finally...a system that won't hold you back.



8. File/Record Locking and Security

When you decide to implement either a network or a multi-user system, there's a two-fold problem which must be solved: protecting your work from accidental misuse and securing it from intentional theft.

PC-MOS/386™ solves both aspects of this problem. Password protected security allows you to assign file, directory, and task access to each user. Plus, files and records are locked using either PC-MOS' proprietary system or NETBIOS emulation.

9. Remote Access



It's been said that information is power...which makes PC-MOS/386™ a deadly weapon to your competition. Imagine on-the-road salespeople being able to file call reports and access your latest inventory data. Picture executives being able to access your corporate database from across the country, or around the world—giving them the information they need, when they need it.

Visualize branch offices tapping time-critical data with nothing more than a modem and a workstation. Working at a home office in the evening or over the weekend suddenly gets awfully productive. And that makes good business sense. The kind of sense you can't afford to be without.

10. The Price...

As you evaluate operating systems, ask yourself if it's reasons you're considering...or rhyme. Ask if you're getting a system for tomorrow, or one that was made for yesterday. See if you're being forced to buy new hardware because of *their* software.

And consider this.

Only one operating system in the world can give you the raw power, features, and functionality that you demand. Its name is PC-MOS/386.™ And it's immediately available in one, five and 25-user versions starting at \$195.

\$195

PC-MOS/386™ is a trademark of The Software Link, Inc. PS/2, PC AT, NETBIOS, dBASE III, MultiMate, WordPerfect, Lotus 1-2-3 & Symphony, & WordStar are trademarks of IBM Corp., Ashton-Tate, WordPerfect Corp., Lotus Development Corp., & MicroPro, respectively. Prices and technical specifications subject to change. Copyright ©1987. All Rights Reserved.

For the dealer nearest you, In Georgia: International/OEM Sales: Resellers/VARs:
CALL: 800/451-LINK 404/441-2580 404/263-1006 404/448-5465

3577 Parkway Lane, Atlanta, GA 30092 Telex 4996147 SWLINK FAX 404/263-6474

The Software Link/Canada CALL: 800/387-0453

DEALER INQUIRIES INVITED



PC-MOS/386™
 MODULAR OPERATING SYSTEM



THE SOFTWARE LINK

impractical as well.

As a result, a package based on a specific piece of hardware will have to include a device driver. Device drivers have access to a set of kernel functions to assist them in translating between virtual and real addresses, in locking storage, in fielding interrupts, and so forth.

Unfortunately, OS/2 device drivers are even harder to write than MS-DOS device drivers. The reason is the uncertainty about the machine state at the time an interrupt occurs. The system might be operating in real-address mode in the simulated DOS 3.3 environment—or it might be operating in protect mode. The interrupt-handling code of a device driver has to be able to operate in either mode. Furthermore, the split between the “strategy” and “interrupt” halves of a device driver, a split that was only formal in MS-DOS, is real in OS/2. The strategy routine has to start, or queue, the work to be done and then get back to the kernel pronto, and the interrupt routine has to be able to operate asynchronously.

The expanded support for the *IOctl* system call might make some device-centered packages easier to design. Application code can exchange data with device driver code through this call. (The distributed OS/2 device drivers support an elaborate scheme of “generic” *IOctl* calls that bring a degree of device independence to this very low level of the system.) If I were designing a package based around a piece of hardware, I would try very hard to put as little function in the device driver as possible and reserve as much as possible to dynalinked code. I'd use the device driver to lock segments in storage and to field interrupts. Everything else could be done from dynalinked code that used the device driver as its private resource.

Dynalink Descriptors

Because I'm into technical details here, I might as well explore a strange addressing problem. This item is merely a sidetrack intended for those who know the 80286 hardware well; others can just skip ahead to the next section.

You might suppose that, as dynalink segments are shared across clients, they must be addressed through the Global Descriptor Table (GDT). Not so. OS/2 appears to reserve the GDT for kernel code and system data objects. Application code segments and dynamic link segments all go in the LDT of each task. It's also possible to share dynamically-allocated data segments between programs, and these shared segments also are addressed through the LDT, not the GDT as you might at first expect.

The reason is probably security. Dynalink code isn't meant to be a global resource; it's linked to a specific client program or programs. Shared segments are only for the use of the clique of programs sharing them. By putting their descriptors in the LDTs of the using programs, the system ensures that only the right programs can use them.

Those who know the 80286 really well will immedi-

ately spot a problem. Loaded code often contains *far* pointers to its own segments, and *far* pointers are constants embedded in the code. It follows that a dynalink segment must use the identical segment address in every LDT in which it is entered. If it didn't, its embedded *far* pointers would be correct for a call from some tasks but not from others.

In order to make that happen, OS/2 has to treat the 8,191 possible entry indexes into an LDT as a global pool—even though an LDT is a private resource of one program. When a dynalink library is first loaded, it is assigned a set of descriptor table entry numbers from the pool. It will use those segment numbers in whatever program's table it appears, and no other segment can use those numbers until this dynalink library loses its last client and is unloaded. There's clearly a potential for creating large, sparse LDTs. It remains to be seen if that will be a problem in production systems.

Errors and Exit Lists

Very well, a client program called your *InitGoody* entry point, then called your *StartGood* entry point to begin a complex operation. And now the client program has done something foolish and has been terminated by the system, leaving your shared data segment or I/O device in a halfway condition, storage allocated, semaphores uncleared. Other clients will suffer. That's not good; it's a principle of OS/2 design that one program's disasters shouldn't affect other programs.

For every program, the OS/2 kernel maintains an exit list—a list of code addresses that want to get control before the program is terminated. There's a system call to enroll an entry point in the exit list. You'd use it from the *InitGoody* entry point, enrolling your *GoodyByBy* procedure as an exit procedure. Now if the client makes a serious mistake, OS/2 will call your code and you can clean up the debris.

The reason it's a list, not just a single address, is that the OS/2 design anticipates that an application might be using several dynalink libraries and every one of them might want to register its own exit procedure. There's no promise about which one will be called first, but that shouldn't matter. Each one should only be concerned with resources that are in its private domain and none should even be aware of the others.

Device Monitors and Replacement Functions

The OS/2 named devices (*PRN*, *LPTn*, *COMn*, *SCREEN\$*, and *KBD\$*) can be opened as files and used via file handles, just as under MS-DOS. Or they can be addressed as device drivers using the generic *IOctl* system calls. The keyboard, screen, and mouse are also supported by dozens of system calls with names such as *KbdCharIn*, *VioWriteCells*, and *MouGetPtr*. All these are directly available to dynalink code, as they are to applications.

There are yet two more levels of control over device I/O. Although available to any code, they're sufficiently complex that they'll probably only be used in dynalink packages. These are device monitors and replacement functions.

A device monitor is a piece of code that monitors the



#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for
FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. \$185.

Special: VEDIT (single file, no windows) for CP/M—\$49.

- Fully Network Compatible
- Call for XENIX-286 version
- 30 Day Money-back guarantee

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20 +	2	No	20 +
Windows	20 +	2	No	20 +
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of				
Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CIRCLE 111 ON READER SERVICE CARD

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

stream of data bytes that flow between a device and a program. The monitor receives a data packet with a call to the *DosMonRead* dynalink function and is expected to pass the packet on with a call to *DosMonWrite*. But there's no rule that it has to pass on every packet or that it pass on the same number of packets. A monitor is free to censor the data stream, to generate new packets, or to substitute packets.

Consider a monitor for the keyboard data stream. It sees all keystrokes, each neatly wrapped up in a "data packet" with flags for the current shift state and a millisecond time stamp. It can interpret the keystrokes as it wishes and substitute for them. In short, a keystroke monitor may be a keyboard "macro" program—and it doesn't have to field interrupts or be loaded in any special way.

A keyboard monitor is created by system calls. A program could set up its own keystroke monitor, but more likely one would be set up in dynalink code (and torn down in an orderly way in an exit procedure). Because it's easy to set one up, there can be more than one. OS/2 permits a whole pipeline of keystroke monitors to exist, each one getting the output of the last and providing input to the next, and they don't have to be aware of each other.

There is a separate logical keyboard for each of the 16 "screen groups," and a keystroke monitor sees only the strokes from the keyboard of one screen group (not necessarily the one its code is running in!). The printer devices, however, are global to all screen groups. A printer monitor sees all the data passing to its printer from all processes in the system. Data packets contain the process ID that produced them, and "open" and "close" packets are visible. This is the basic function needed to build a print spooler: a program that intercepts multiple data streams, saves the data on disk, and sends files on to the printer in orderly sequence. An extremely rudimentary spooler is built on this facility and distributed with OS/2, but there is plenty of scope for better ones to be built. There are intriguing possibilities for building completely transparent support for network printers and for building filters that translate printer control codes for one make of printer to another's. All of this happens, remember, at the level of application code, without any requirement for I/O privilege or hardware dependency.

The dynalink libraries distributed with OS/2 contain, as I mentioned, dozens of system calls for using the keyboard, the mouse, and the screen. The screen calls contain all that's necessary for a full-screen editor, for example.

The distributed code, however, assumes the presence of an IBM-compatible keyboard, an IBM-compatible screen adapter, and one of a small number of mouse devices. That isn't enough. The OEMs that sell OS/2 systems will have the know-how and development tools to build replacement device drivers and dynalink libraries that support their particular hardware under the standard calls. But there will be times when the code for

a keyboard, mouse, or screen operation should be replaced at the level of an individual program or an individual screen group (multiple programs may run in a single screen group).

This need is provided for. It is possible for a program to "register" a replacement procedure for almost any of the supplied mouse, screen, or keyboard calls. From the moment of registration, whenever the supplied function is called, control will be transferred to the registered replacement. That procedure gets the same stacked parameters as the original function would have seen. It may choose to interpret them in a different way (permitting access to a wider screen, for example), or apply them to the hardware in a different way, or censor or expand on them in the manner of a device monitor, or perhaps just record them for performance monitoring. The Presentation Manager will probably use this facility to take over the mouse and screen calls in the screen group that it controls.

Run-Time Linking

I alluded to the ability to do a very late dynamic link, after the program has been loaded and while it is executing. There is a system call that will take the file name of a dynalink library and load it, returning a handle. There is another call that takes such a handle and the character name of an entry point and returns a far pointer to the entry point. The calling program may then call the dynalink entry point.

This isn't genuine linking because calls to the external code can't be freely embedded in the caller's code. Calls have to be indirect by way of pointers in storage, and the linkage must be established with the explicit system calls. Still, it's a facility with some important uses. Basically, it gives a program the ability to configure its own contents at run time. The components of a large subsystem may be linked independently, each as a dynalink library. The kernel of the subsystem may decide at execution time which of its components ought to be loaded. A communications subsystem, for example, might dynamically load its components for different line protocols in this way, as needed.

Coding Dynalinks

Between dynamic linking, exit lists, device monitors, and replacement device functions, it ought to be possible to extend OS/2 in about any direction. Some extensions are more obvious than others. For instance, a dynamic link library would appear to be about the ideal way to package a compiler's run-time library. Think what a reduction that would make in the size of compiled programs! Unfortunately, it looks as if the languages released with Version 1.0, at least, will have their run-time code in the old-fashioned kind of object library.

Something else obvious is that it would be very desirable to write dynalink modules in high-level languages. Here, alas, you run into a major glitch in the system design. The code generated by the IBM/Microsoft C, Pascal, and FORTRAN compilers is not suitable for dynamic linking!

The reasons aren't too hard to understand. A dynamically linked module can be entered from different pro-



Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

Step Lively

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

Cut Corners

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered, interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

Get on the Fast Track

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

Greenleaf Comm Library	\$185.00
Greenleaf Functions	\$185.00
Greenleaf DataWindows	\$225.00
Greenleaf C Sampler	\$ 94.50
Digiboard Comm4	\$325.00
Digiboard Comm8	\$535.00

In stock, shipped next day.



Greenleaf DataWindows and Turbo C

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And, our new 3-in-1 C Sampler for only \$94.50 supports both Turbo C and Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.



GREENLEAF
Software

Call Toll Free:

800-523-9830

In Texas and Alaska:

214-446-8641

Greenleaf Software, Inc.
16479 Dallas Parkway, Suite 570
Dallas, Texas 75248

grams. It has to be prepared to use its caller's stack, and it cannot make any assumptions about the way its caller uses registers. Unfortunately, the code from most compilers *does* make assumptions about registers, lots of them. The IBM/Microsoft compilers are particularly dependent on two assumptions that just aren't true on entry to dynalink code: that *ss=ds* and that *ds* bases *DGROUP*.

Neither of these is a big problem. The assumption that the stack and data segments are identical was a lazy coding ploy (don't bother saving *ds*, just reload it by pushing *ss* and popping *ds*) that ought never to have been allowed. This assumption isn't true in large-model C anyway, so it shouldn't be too difficult to eradicate it from all the code generators and all the run-time libraries.

The assumption is that *ds=DGROUP* is more subtle. Every compiled module has a *DGROUP*, a segment of static data. When object modules are linked, the linker merges all *DGROUPs* into one segment and adjusts the offsets in all instructions as required. This works just fine for an application program because all its parts are linked in one run. Whatever object code it includes references the same *DGROUP*. But the code of a Dynalink library is linked in a separate run. The pieces of its *DGROUP* are formed then, and its *DGROUP* is nothing like the *DGROUP* formed when its clients are linked.

But a compiler's code generator may assume that there is only one *DGROUP* and that *ds* addresses it at all times after the initialization of the module. That ain't true when a dynalink procedure is entered; then *DS* addresses the caller's *DGROUP*. What's lacking is that the prologue to any public procedure should contain the sequence familiar to any assembly-language programmer:

push ds

```
mov ax, seg DGROUP
mov ds,ax
assume ds:DGROUP
```

And, of course, the exit code of any public procedure would have to contain *pop ds* to restore the register.

Current compilers do not generate this code, and as a result the code they produce can't be used as dynalink procedures! Not, at least, unless it has an assembly-language front end to switch *DGROUPs* for it.

One other irritating problem gets in the way of high-level dynalinks. Existing compilers are prone to generating automatic checks of one kind or another—for stack overflow, for subscript ranges, for whatever. Not all of these can be disabled. If even one of them is left, some kind of error-handling module will be included in the link. But the only way such modules have to handle an error is to issue a message to the console, and the only way they can seem to find to issue a console message is to use the compiler's file I/O mechanism, so the presence of even one stack check causes a major part of the compiler's run-time library to cascade into the dynalink code. And that often brings with it modules that won't link properly in a dynalink.

Microsoft has been thoroughly beaten up on for these problems at developers' conferences and claims to be working on solutions. It'd better be working hard. Dynamic link libraries are an extremely attractive facility of OS/2, and the first programming languages that support them properly will have a competitive edge in a system that (Bill Gates confidently says) will have ten million users by 1992.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 1.

PERISCOPE™

POWER

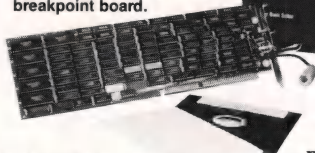
...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.

... Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus \$10! And don't worry about having a lot more to learn ... Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write ... thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Top-of-the-line Periscope III with real-time, hardware breakpoint board.



Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for \$345.

Periscope II includes break-out switch; software and manual for \$175.

Periscope II-X includes software and manual (no hardware) for \$145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is \$995; for machines running up to 10 MHz, \$1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED.

800-722-7006

The
PERISCOPE
Company, Inc.

1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

CIRCLE 113 ON READER SERVICE CARD

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF

into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFO WORLD AND PC MAGAZINE.

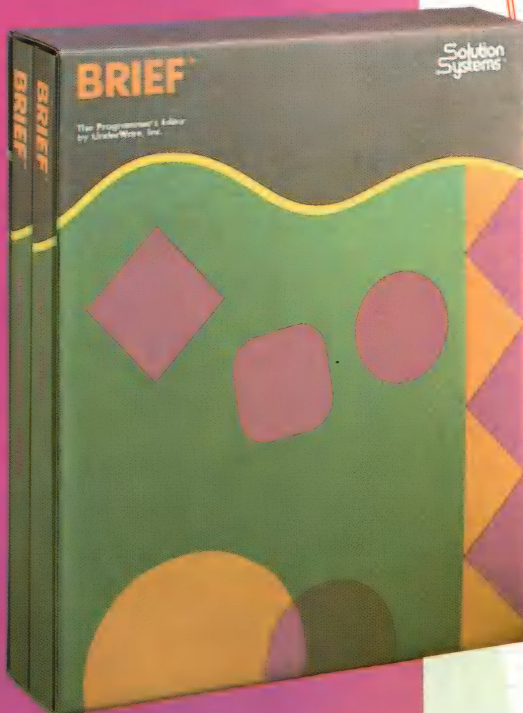
One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. BRIEF 2.0 is just \$195. If you already own BRIEF, call for upgrade information.

**TO ORDER CALL: 1-800-821-2492
(in MA call 617-337-6963)**

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

**Solution
Systems™**

541 Main Street
Suite 410D
So. Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions and the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic
features that made
BRIEF SO popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

Requires an IBM PC or compatible with
at least 192K RAM.
BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

A RAM-Cache Manager in C

by Alan Deikman

To an application program in most operating systems, a read/write request appears to take place instantaneously—throughput is primarily dependent on the I/O speed of the devices being accessed. For some devices this is an acceptable limitation. After all, during the interval a program is waiting to receive user input from the keyboard, there usually isn't anything for the program to do but wait.

For random-access disk requests, however, the data being sought, which may not arrive for another 80 milliseconds or so, may actually have been in-hand only moments before. If a program had the intelligence and foresight to save that block of data in RAM, there might not be any need for disk access at all and, as such, no need for that 80-millisecond delay.

When accessing disk blocks, it's a common practice to access the same blocks over and over again and to access other blocks only once in a long while. The operating system portion of the disk is typically the most often accessed—in MS-DOS, this section holds the directories and the file allocation tables (FATs).

In a multiuser operating system, ideally, CPU time is diverted to work on some other task when a program requests a disk block, with control passing back to the primary pro-

Prevent disk bashing with RAM caching

gram as soon as the required data becomes available. Overall system throughput is thus enhanced at the expense of the individual program. Even so, there's no advantage in letting an application program force the operating system to access the disk more than is necessary.

If such a program is used frequently, the capacity of the entire system can be greatly increased if each program is optimized for fewer time-intensive disk requests. For I/O-intensive applications, this becomes a critical issue. An accounting or database system does very little computation as a rule, and most of the time spent (after the user has entered input) is spent waiting to store or retrieve data.

Within this context, there are two primary options for increasing program performance:

1. Buy faster hardware.
2. Reduce the number of actual disk accesses required.

Disk caching is one of the least expensive solutions available to the programmer to optimize program performance.

Cache Theory

To make a disk cache, an area of RAM is set aside to hold the most recently used blocks of data. Each

block is identified by block number. Whenever a request for a disk block read occurs, the cache area is checked first to see if the required block is available. If it is available, no disk access occurs, and the calling program uses the RAM copy of the block. That disk block is then tagged "most recently used" (MRU). When this happens it is called a cache hit.

If it turns out that a disk access is required (because a cache miss occurred), the block is obtained from the operating system as usual. Before the block is used, however, it's copied into the cache area, replacing the "least recently used" (LRU) block in the cache area, if any exists. It is then marked MRU.

When a disk block is written, it also is placed in the cache as the MRU block. In most applications it is desirable to write the block physically to the disk at that time, but sometimes it is better to wait until the block becomes the LRU block and is about to be overwritten. It is possible that the block needn't be written in the first place.

The routines in this article maintain a two-way linked list, called the LRU/MRU chain. The *cacallo()* routine sets up the original linked list to include all the blocks allocated, as illustrated in Figure 1, page 31. A -1 (0xFFFF) flags the end of the chain.

Whenever a block is designated the MRU, it is taken out of the linked list and reinserted at the end. This operation is performed by the function *cacnew()*. If this operation were to be performed on block 2 of the initial chain, the result would be as shown in Figure 2, page 31.

Alan Deikman, Software Services, Menlo Park, CA 94026-2106. Alan is a management, marketing, and computer consultant. He has been consulting since 1978 and has had an independent practice since 1985.

Cache Size and Application

There's a trade-off between cache size and net efficiency. If the cache is too small, the likelihood of hits is small. On the other hand, if the cache is too large, the CPU spends more time than necessary looking up disk blocks in the cache. (The MS-DOS manual warns against this, in the discussion on setting up the *BUFFERS* command in *CONFIG.SYS*; the *BUFFERS* command sets up a system cache of 128-byte records.)

Figure 3, below, shows the curve representing an application that uses caching with perfectly random disk accesses. The left side of the curve represents no caching, and for small caches a decrease in performance occurs. Performance increases to some theoretical optimum and then falls off. Diminishing returns ultimately make disk caching more of a burden than a benefit.

Most applications, however, are not truly random in the way in which they access the disk. There are cases in which disk caching is misapplied. Consider the situation in which a file is read from beginning to end. No block is ever read twice, so applying disk caching is merely adding overhead. If the file is one block larger than the cache area, and must be rewound and read again, then each read on the second pass will result in a cache miss. Thus it is never desirable to apply disk caching to sequentially read files of an unknown size.

In yet another case, when the cache memory is bigger than the disk file, disk caching becomes superfluous because it would have been better to read the whole file into RAM initially, then access the blocks directly, than to put up with the storage overhead of keeping MRU/LRU information.

Because there are so many variables associated with the efficiency of caches, I've provided the *cacstat()* routine to obtain the statistics of the cache, allowing the user to adjust the size of the cache for optimum performance. The values returned are cache hits, cache misses, and cache adds. The sum of hits and misses represent the total number of times the cache was searched for a block. The optimum ratio of hits to total accesses depends on a

number of variables such as the ratio of CPU/RAM speed to disk average access time.

Multiple Caches

One of the most glaring failings of having an operating system perform all the cache work for disk I/O is that the operating system isn't in a posi-

tion to discriminate between different types of accesses. Also, most systems can't allocate cache memory on a dynamic basis. As a result, the operating system cache can be wiped clean of useful records just because the application made one pass of an input file, filling the cache with records that are

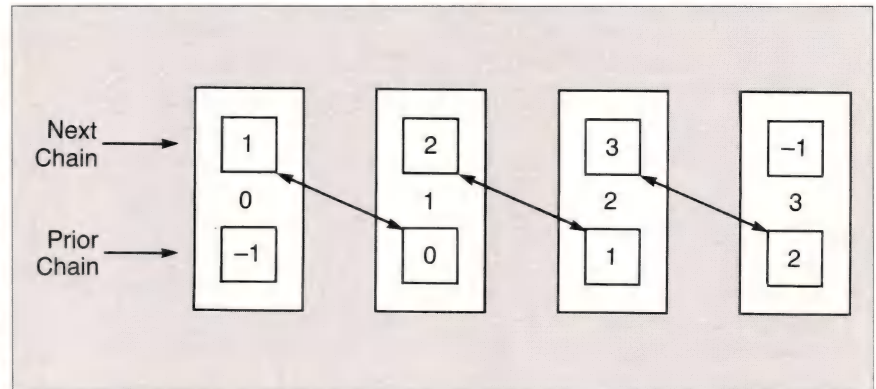


Figure 1: Initial LRU/MRU chain

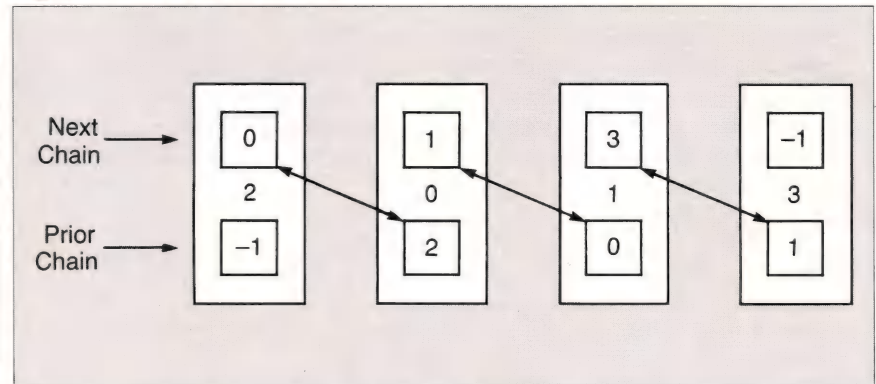


Figure 2: After block 2 is made MRU

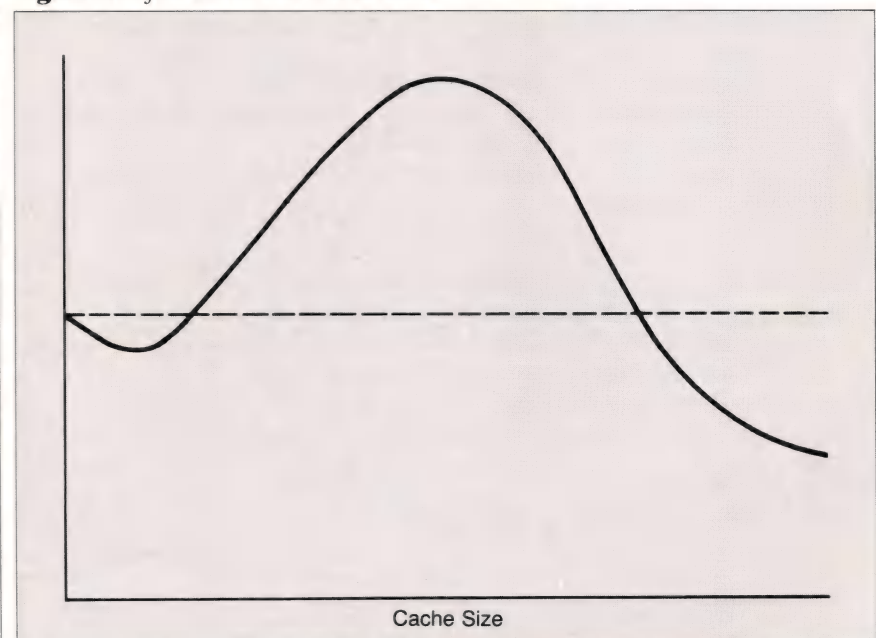


Figure 3: Cache size vxs. performance

New from **MKS**

MKS RCS

Revision Control System

With **MKS RCS** running under DOS, programmers, systems administrators, project managers and librarians can efficiently control and record the revisions of text files such as programs, documentation, graphs, papers, form letters, and so on. This software package:

- resolves access conflicts — for example, it prevents two programmers or authors from making simultaneous changes to a file;
- maintains a complete history of changes, including date and time of change, author, and reason for the change;
- allows retrieval of any version of the file, by date, release number, or a user-assigned name;
- runs quickly because only the most recent version of a file is stored — non-current versions are stored as difference-files to minimize storage requirements;
- allows divergent versions to branch from the main family of versions — these might be test versions of programs, or customized versions of documents;

Programs included in the package:

- **ci** — check in **RCS** revisions
- **co** — check out **RCS** revisions
- **ident** — display **RCS** identification information
- **merge** — three-way file merge
- **rcs** — change **RCS** file attributes
- **rcsclean** — clean up work files
- **rcsdiff** — compare **RCS** revisions
- **rcsmerge** — merge **RCS** revisions
- **rlog** — display log messages about **RCS** files
- **diff** — show minimal file differences
- **diff3** — show differences among three files

The entire system for **\$189.**

Also available:

The **MKS Toolkit**: over 110 UNIX-based tools for DOS including the **Korn Shell**, **Vi**, and **AWK**, complete with nearly 400 pages of documentation and tutorials. The complete package: **\$139.**

AWK: The 4th generation language fully compatible with the latest description in *The AWK Programming Language*, by Aho, Weinberger, and Kernighan. **AWK** with tutorials and documentation: **\$75.** Both the software and *The AWK Programming Language*: **\$89.**

MKS Vi: the UNIX screen editor running under DOS at lightning fast speeds — it's tuned for the PC. Comes with Tutorial and Reference Manual for **\$75.**

Mortice Kern Systems Inc.,

43 Bridgeport Road East, Waterloo, Ontario, Canada, N2J 2J4 (519) 884-2251

uucp: {allegro, decvax, ihnp4}!watmath!mks!toolkit

MKS RCS runs under MS-DOS 2.0 or later. Not copy protected. Prices quoted in US funds. VISA, MASTERCARD, American Express, uucp, and purchase orders are accepted. Overseas orders please add \$10 for postage and handling. MKS and MKS RCS are registered trademarks of Mortice Kern Systems Inc. UNIX is a trademark of AT&T Bell Labs. MS-DOS is a trademark of Microsoft Corp.

RAM-CACHE MANAGER (continued from page 31)

never read again. If the system is multitasking (or multiprocess), the situation would be even worse because other programs could dominate the available cache space.

It's usually obvious to the program designer which disk files should be cached and which shouldn't. A good application for a disk cache is a compiler's temporary tables, where using a disk scratch file is considered only after RAM memory runs out. In B-tree indexing subroutine libraries, caching is particularly effective in processing look-ups and node additions.

For this reason, all the routines provided in this article operate on a structure that is pointed to by a single variable kept by the calling program. This approach allows any number of separate caches of variable sizes to be managed concurrently.

The *cacallo()* routine uses the standard library *malloc()* routine to allocate all memory necessary for the cache. It returns a pointer to the root structure of the cache that is used as a parameter to all the other cache routine calls. All the global parameters, and pointers to other objects, are contained in this structure, which is *typedefed* to be *CACDS*.

Four parameters are required to set up a cache:

- the number of records in the cache
- the length of each record
- a pointer to an external function for processing free records
- an identifier word (*long*) to pass to the external function

If the application interface is not going to defer the writing of disk blocks, a free record-processing routine is not necessary. In this case, the third parameter provided should be a null pointer (*char **) 0. The identifier word is used when a single routine is being used to handle the freed blocks from multiple caches. This value can be used by the called routine to identify from which cache the record is being transmitted.

Once again,
Compaq
raises the standard
of performance
for personal computers.

This time
by a factor of two...



Introducing the two on earth



The new COMPAQ DESKPRO 386/20™

Last year, we introduced the COMPAQ DESKPRO 386™ the most advanced personal computer in the world. Now the world has two new benchmarks from the leader in high-performance personal computing. The new 20-MHz COMPAQ DESKPRO 386/20 and the 20-lb., 20-MHz COMPAQ PORTABLE 386 deliver system

performance that can rival minicomputers. Plus they introduce advanced capabilities, without obsoleting your investment in software, hardware and training.

Our new computers employ an industry-standard 20-MHz 80386 microprocessor and sophisticated 32-bit architecture.

But to make these two of the world's fastest PC's, we did more than just increase the clock speed.

For instance, both are built around a concurrent bus architecture. Two buses—one for memory and one for peripherals—eliminate information bottlenecks, allowing each component

It simply works better.

most powerful PC's and off.



and the new 20-MHz COMPAQ PORTABLE 386™

to run at its maximum speed. Together, they insure the highest system performance without sacrificing compatibility with industry-standard peripherals.

Both computers offer disk caching. Both offer the most memory and storage within their classes. Both let you run software being written to take ad-

vantage of 386 technology. And both run new MS-DOS®/BASIC Version 3.3 as published by Compaq. With it, our new portable and our new desktop can break the 32-megabyte limit on file sizes that handcuffs other PC's, allowing you to build files up to the size of your entire fixed disk drive.

And from now until December 31, 1987, both computers come with a free package of new Microsoft® Windows/386 Presentation Manager. It provides multi-tasking and switching capabilities with today's DOS applications to make you more productive. But that's just the beginning. To find out more, read on.

COMPAQ®

The question wasn't but how to get the

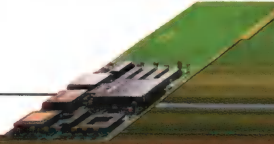
System Board with 20-MHz Cache Memory Controller



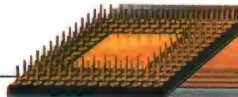
135-MB Tape Backup



Weitek Coprocessor Board



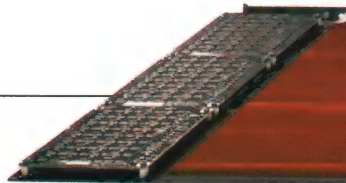
20-MHz 80386 processor



300-MB Fixed Disk Drive



16 MB of 32-bit RAM



The most powerful personal computer in the world

The COMPAQ DESKPRO 386/20 is an impressive 50% faster than 16-MHz 386-based personal computers. Even more impressive is the fact that it's up to 25% faster than other 20-MHz 386's. That's because the processor is just one small part of how the COMPAQ DESKPRO 386/20 outperforms every other PC

in the world today and even many minicomputers.

The big reason is the new COMPAQ Flexible Advanced Systems Architecture, which optimizes overall system throughput while maintaining full compatibility with industry-standard peripherals. It does this by combining an

advanced memory caching scheme with memory and peripheral buses that operate concurrently.

Complementing the speed of the microprocessor is the new advanced 20-MHz Intel® 82385 Cache Memory Controller. Like an efficient secretary that keeps frequently used information close at hand, it allows the microprocessor to operate at 0-wait states 95% of the time.

While one bus handles these high-speed operations, another *simultaneously* handles periph-

It simply works better.

how to get to 20 MHz, most out of 20 MHz.



erals operating at the industry-standard 8 MHz.

This flexible approach allows you to dramatically increase system throughput while preserving your investment in monitors, disk drives, and expansion boards. It can also accommodate today's and tomorrow's most advanced peripherals without constraining their performance.

Take options like our new Weitek™ Coprocessor Board. Never before offered in a PC, it can increase the speed of calculation-intensive, engineer-

ing and scientific applications by a factor of six, giving the COMPAQ DESKPRO 386/20 the performance of a dedicated engineering workstation at a fraction of the cost.

Compaq also provides 130- and 300-Megabyte Fixed Disk Drives with some of the industry's fastest access times. And when used with disk caching software, they represent the highest-performance storage subsystems available.

As for memory, Compaq offers 32-bit high-speed RAM.

One full megabyte comes standard and is expandable to 16 megabytes without using an expansion slot. Plus, we included the COMPAQ Expanded Memory Manager. It supports the LIM standard so your software can break the 640-Kbyte barrier even before OS/2™ is released.

As tasks become more complex and users demand more advanced capabilities, Compaq responds by raising the standard of performance in personal computing.

COMPAQ
DESKPRO 386/20™

Everyone expected Compaq But no one



Pound for pound, it is the world's most powerful computer

Compaq has long been recognized as the world leader in both 80386 technology and portable computing. So it isn't surprising that we would combine the two.

But no one expected the new COMPAQ PORTABLE 386 to run at 20 MHz. And no one even

dreamed that it would offer 100 megabytes of storage, disk caching, and much, much more.

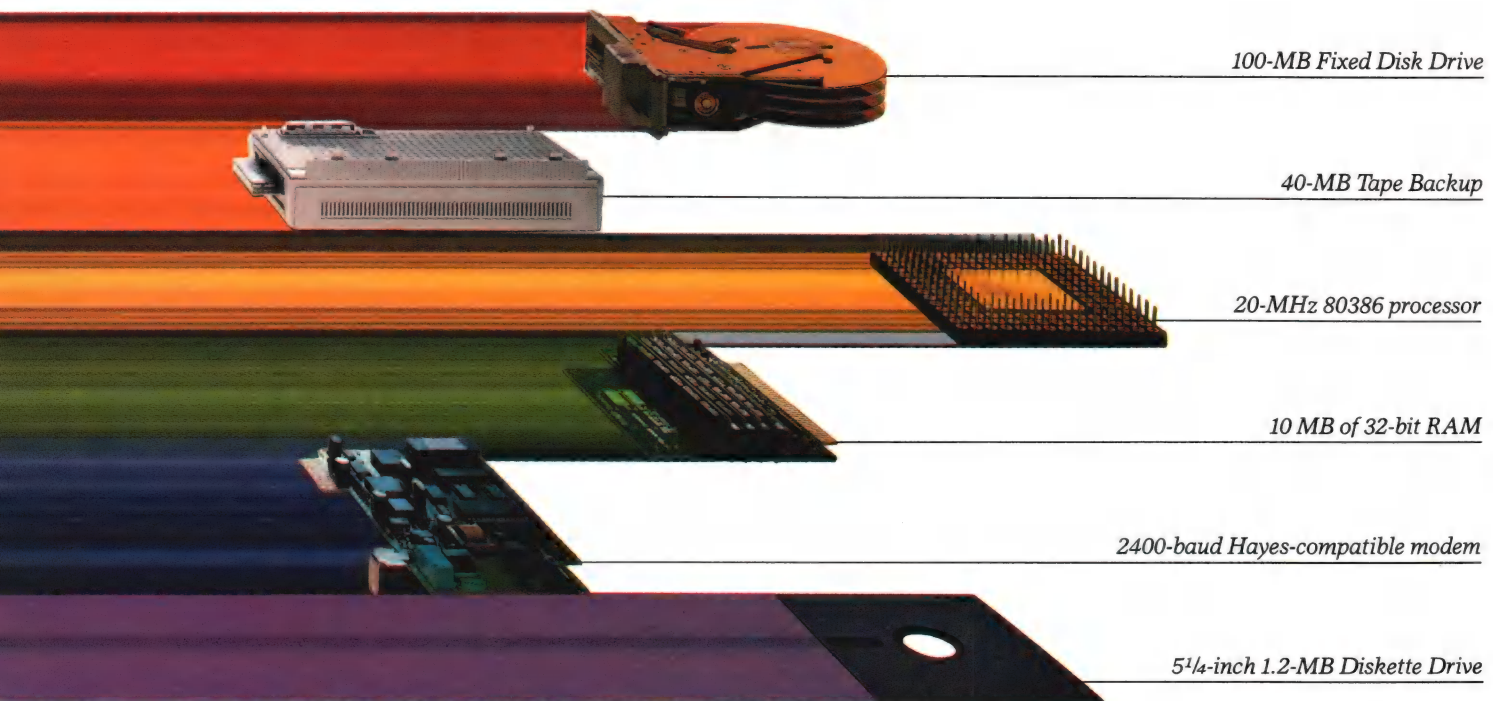
Our newest 20-lb. portable computer goes far beyond an 80386 microprocessor with a handle. It's not just the most advanced portable in the world.

Pound for pound, it's the world's most powerful computer. Period.

Like the recent COMPAQ PORTABLE III™ which changed the shape of full-function portable computing, the COMPAQ PORTABLE 386 makes no compromises. It offers more speed, memory, storage and features than any other portable PC. It runs your current software up to 25% faster than 16-MHz 386 PC's. Beyond that, its performance in calculation-intensive

It simply works better.

to introduce a 386 portable PC. expected all this.



100-MB Fixed Disk Drive

40-MB Tape Backup

20-MHz 80386 processor

10 MB of 32-bit RAM

2400-baud Hayes-compatible modem

5 1/4-inch 1.2-MB Diskette Drive

applications is increased even more when you add an optional 20-MHz 80387 coprocessor.

Memory? Get one megabyte of 32-bit, high-speed RAM standard or go as high as 10 MB internally. And like all of the COMPAQ 386-based PC's, it features the COMPAQ Expanded Memory Manager.

With our high-performance 100-megabyte internal fixed disk drive, you can actually fit 500 lbs. of data-filled pages into a 20-lb. PC,

unsurpassed storage for a portable. If that's too much for you, we also offer a 40-megabyte model.

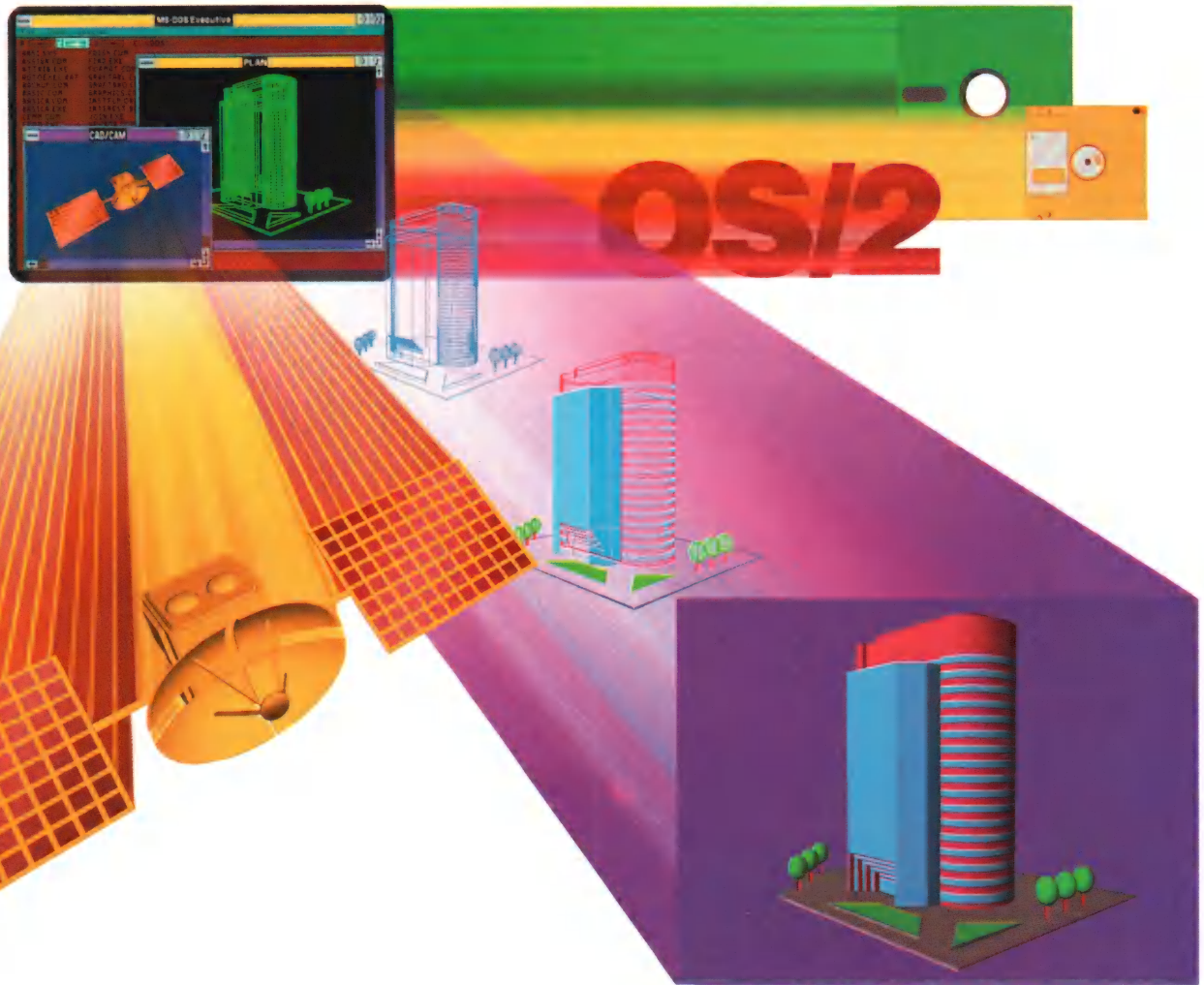
We've become famous for building desktop computer capabilities into our portables without leaving anything out. The COMPAQ PORTABLE 386 is more proof. It has a high-resolution, 640 × 400, 10-inch plasma display; a full-size, portable enhanced keyboard; two industry-standard expansion slots in a lightweight, optional plug-on unit; a choice

between an optional 2400- or 1200-baud Hayes-compatible modem; a full-size 5 1/4-inch 1.2-MB diskette drive; even an optional 40-MB tape backup.

These features, combined with the ultimate in portable performance, make the COMPAQ PORTABLE 386 the *biggest PC this small.*

COMPAQ
PORTABLE 386™

Compaq moves you ahead without leaving you behind.



Compaq offers the most complete line of high-performance 386 solutions. They all run industry-standard software and hardware, protecting the investments you've already made.

At the same time you won't be left behind when other technologies become important. Multi-task with existing applications using Microsoft Windows/386 Presentation Manager. Add VGA

graphics if you wish. Run OS/2 when it's available. And now 3 $\frac{1}{2}$ -inch drives are even an option for our desktops.

We optimize the most advanced technology while maintaining compatibility with the past, present and future. This makes COMPAQ PC's a wise decision for serious business users. Because at Compaq, we don't burn bridges, we build them.

See the COMPAQ DESKPRO 386/20 and COMPAQ PORTABLE 386 at an Authorized COMPAQ Computer Dealer. And from now through December 31, 1987, get Microsoft Windows/386 Presentation Manager free when you buy a 386-based COMPAQ computer. For more information, call 1-800-231-0900, Operator 40. In Canada, call 416-733-7876, Operator 40.

Weitek™, Lotus® Intel®, Microsoft® MS-DOS® Hayes® and OS/2™ are trademarks of their respective companies.
©1987 Compaq Computer Corporation.
All rights reserved.

COMPAQ®

It simply works better.

Using the Cache Routines

All the routines and the *typedef* for the cache structure are in the file *cache.h*, shown in Listing One, page 62. It's not really necessary for the calling routine to have access to *typedef* because the calling routines never need direct access to the *CACDS*.

Cache.h declares the structure in *typedef*, however, so that the calling routine can conform to ANSI specifications by declaring an external or local variable of type *CACDS*. (With most compilers it is acceptable for the calling routine to store the returned value of *cacallo()* in a character pointer.)

The *cache.h* file also provides the function declarations necessary for function type checking by the compiler. Listing Two, page 62, shows all the cache-processing routines, which may be compiled separately and/or incorporated into a subroutine library. Listing Three, page 67, is a simple test program for these routines.

After the cache has been set up, it will have to be checked to see if the desired record is already stored there. To do this, the *cacfind()* routine is called. If the cache record designated by the *num* parameter isn't found, a null pointer is returned and the calling routine can then take the appropriate action. Usually this means issuing a *read()* request to get the desired block, then adding the record to the cache according to the procedure that follows.

If the cache record is found by *cacfind()*, a character pointer to the desired record is returned to the calling routine. That record is also taken out of its current position in the LRU chain and added at the MRU end. For example, suppose the calling routine is accessing blocks of 512-byte records and wants to read the record whose number is stored in the variable *recn*. The external character pointer *cache* is initialized with the return value of an earlier call to *cacallo()*. The code (sans any error checking) might appear as shown in Example 1, right.

This example doesn't add any records that were read from the disk to the cache. A record produced by the *read()* call should be added to

the cache at the MRU end because, after all, it's the most recently used record. Each time a record has to be added to the cache, the *cacnum()* routine is called. It is im-

portant to note that the *cacnum()* routine does not check to see if the added record is already in the cache. Thus each call to *cacnum()* should be preceded by a *cacfind()*

```
char buffer[512];
long recn;
char *rec;

{
    if ((rec = cacfind(cache, recn)) == NULL) {

        /* record not in cache, must be read from disk */

        lseek(fd, recn * 512, 0);
        read(fd, buffer, 512);
        rec = buffer;
    }

    /* rec points to record to process */
}
```

Example 1: Accessing blocks of 512-byte records

```
char buffer[512];
long recn; /* record number to write */
char *rec; /* pointer to block in cache area */

{
    if ((rec = cacfind(cache, recn)) == NULL) rec = cacnum(cache, recn);
    memcpy(rec, buffer, 512); /* copy data into cache area */
    lseek(fd, 512 * recn, 0);
    write(fd, rec, 512);
}
```

Example 2: Writing blocks of 512-byte records

NEW!

Documentation is a PAIN!

... without **DocuMotion™**

We've found that well indexed and easily accessed documentation is a powerful tool and asset. Now you can simply pop up **DocuMotion** to access, display and print your documentation. **DocuMotion** builds indexed document libraries from documentation contained in your source code or any text file.

DocuMotion for programmers:

- Your documentation is available ANYWHERE, ANY TIME.
- Access, display and print your documentation by name or by user-defined categorization trees.
- 19 function Microsoft Windows-style menu bar.
- Powerful print & copy functions.

DocuMotion for project mgrs:

- Programmers produce more and better documentation.
- Reduced function redundancy.
- Greater programmer productivity.

DocuMotion for the PC:

- Runs memory resident or non-resident on any IBM PC/XT/AT or compatible.
- Compatible with all popular memory resident programs.
- Requires only 93K RAM.
- LAN compatible.

DocuMotion is for YOU:

Call NOW 1-612-884-5860

At a special introductory price of **ONLY \$159.95** with ANSI 'C' document library.

Demo disk for \$10.00 that puts the ANSI 'C' library functions on line.

NWP - Intelligent Solutions, Inc. P.O. Box 20478 Bloomington, MN. 55420-0478

CIRCLE 116 ON READER SERVICE CARD

RAM-CACHE MANAGER

(continued from page 33)

call.

The *cacnum()* routine does the following:

1. Finds the LRU record in the cache area.

2. If that record has been marked for processing, calls the external free block-processing routine.
3. Makes that record the MRU.
4. Numbers that record with a new number supplied by the calling routine.
5. Returns a character pointer to that record.

```
char buffer[512];
long recn; /* record number to write */
char *rec; /* pointer to block in cache area */

{
  if ((rec = cacfind(cache, recn)) == NULL) rec = cacnum(cache, recn);
  memcpy(rec, buffer, 512); /* copy data into cache area */
  cacproc(cache, recn); /* mark the block for processing */
}
```

Example 3: Code to defer writing records

```
write_cache(idnt, recn, recb)
long idnt; /* cache identifier */
long recn; /* record number */
char *recb; /* record buffer */
{
  lseek(fd, recn * 512, 0);
  write(fd, recb, 512);
  return;
}
```

Example 4: Example function to write a record

To avoid having to copy blocks from place to place in RAM, and to eliminate the need to allocate an external buffer, the calling routines should access the records through a character pointer returned by *cacfind()* and *cacnum()*.

To complete the previous example for reading records, the following code (again without error checking for simplicity) can be used:

```
long recn;
char *rec;

{
  if ((rec = cacfind(cache, recn)) == NULL) {

    rec = cacnum(cache, recn);
    lseek(fd, recn * 512, 0);
    read(fd, rec, 512);

  }

  /* rec points to record to process */
}
```

Writing data records undergoes a similar process. A record that is written also becomes the MRU record via the *cacfind()* and the *cacnum()* routines. If the record is already in the cache, however, there is no need to call *cacnum()*. The complementary routine to the previous one might look like that shown in Example 2, page 33.

Processing Freed Records

As mentioned earlier, it is sometimes advantageous to enter outgoing records (those to be written to disk) into the cache without actually performing the disk I/O operation, then write them to disk only when the block within the cache that the record is sitting on needs to be used for some other purpose. Typically, this would be the case for a scratch file (such as a symbol table), where the file would not even be created unless the data overflowed the cache. In most other cases, it is best to write the data records immediately, as in Example 2.

If the writing of data records is to be deferred, the third and fourth parameters to the *cacallo()* function are used and the *cacproc()* function is used to mark records that are to be processed before the space they

New! Hire a Pro for Your New Turbo 4.0

Turn on the power of Turbo PROFESSIONAL 4.0, a library of more than 300 state-of-the-art routines optimized for Turbo Pascal 4.0. You'll have professional quality programs finished faster and easier.

Turbo PROFESSIONAL 4.0 includes complete source code, comprehensive documentation and demo programs that are powerful and useful. The routines include:

- Pop-up resident routines
- BCD arithmetic
- Virtual windows and menus
- EMS and extended memory access
- Long strings, large arrays, macros, and much more.

Turbo PROFESSIONAL is only \$99.

Call toll free for credit card orders.

1-800-538-8157 extension 830

1-800-672-3470 extension 830 in CA

Satisfaction Guaranteed or your money back within 30 days.

Turbo Pascal 4.0 is required. Registered owners of Turbo Professional by Sunny Hill Software may upgrade for \$30. Include your serial number.

For other information call 408-438-8608, 9 AM to 5 PM PST. Shipping & taxes prepaid for US and Canadian customers, others please add \$6 per item.



TurboPower Software 3109 Scotts Valley Dr., Suite 122 Scotts Valley, CA 95066

CIRCLE 117 ON READER SERVICE CARD

The Software Family

Dept DD127, 649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

**Trust TSF to provide the
best value and service!**

- Technical advice and support by programmers
- Honest and equitable business practices
- Prompt, flexible service to meet your needs

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over \$100 (\$3 delivery charge on orders under \$100). We add actual charges for UPS Blue Label or Federal Express rush service. Our COD fee is \$2. We allow return privileges on most products. We carry a large inventory and ship promptly, so you receive your shipment within 3 to 6 working days of placing your order. Give us a try!

Borland Specials

Turbo C v1.1 (list \$100).....	\$59
Turbo Pascal v4 (\$100) <i>New Version!</i>	\$59
Pascal Editor Toolbox (list \$100).....	\$59
Pascal Database Toolbox (list \$100)....	\$59
Pascal Graphix Toolbox (list \$100)	\$59
Pascal Numerical Methods (list \$100)	\$59
Turbo Basic (list \$100).....	\$59
Basic Editor Toolbox (list \$100).....	\$59
Basic Database Toolbox (list \$100).....	\$59
Basic Telcom Toolbox (list \$100)	\$59
Turbo Proglog (list \$100).....	\$59
Prolog Toolbox (list \$100)	\$59

Prices good until December 31, 1987

**Basic
Turbo Pascal
Publishing**

**C
dBase
AI**

C

<i>Blaise C Tools +</i> (list \$175)	\$125	
<i>Blaise Turbo C Tools</i> for Turbo C (list \$129)	\$95	<i>New!</i>
<i>Creative Vitamin C</i> Specify compiler (list \$225)	\$159	
<i>Creative Programming VC Screen</i> (list \$100)	\$79	
<i>Gimpel PC-Lint</i> (list \$139)	\$103	
<i>Matrix Synergy Development Toolkit</i> (list \$395) ...	\$309	<i>New!</i>
Macintosh-like icons, menus, multi-tasking, mouse, more!		
<i>Metagraphics Metawindow/All Language</i> (\$195)...	\$159	
Windows, graphics, mouse support		
<i>Metagraphics Metawindow/Turbo C</i> (\$95)	\$79	
<i>Microsoft C v5 w/Codeview & Quick C</i> (list \$450)	\$279	<i>New!</i>
<i>Microsoft Quick C</i> (list \$99)	\$68	<i>New!</i>
<i>Softfocus "C" BTree & ISAM w/source</i> (\$115)	\$105	
<i>Sterling Castle Blackstar "C" Functions</i> (\$129).....	\$99	<i>New!</i>
Full featured library w/source for both MS & Borland		
<i>Unipress Phact RDBMS</i> (\$249)	\$199	<i>New!</i>
includes report writer, query program & all source		

TURBO PASCAL

<i>Blaise Turbo Power Tools +</i> (list \$99)	\$79
<i>Metagraphics Metawindow/Turbo Pascal</i> (\$95)	\$79
<i>MicroHelp Mach2 for Turbo Pascal</i> (\$69)	\$55
windows, fast i/o, critical error trapping, more	
<i>Turbo Power TDebug Plus v2</i> (list \$60)	\$48
<i>Turbo Power Turbo Utilities w/source</i> (\$95).....	\$76

dBASE

<i>Bytel Genifer</i> dbase III application generator (\$395)	\$225
<i>Comtel dbScope</i> Interactive dBase debugger (\$70)	\$62
<i>Comtel dbTools</i> (list \$70)	\$62
<i>Fox FoxBase + v2</i> fast dBase compiler (\$395)	\$249
<i>Tom Rettig Library</i> specify dBase or Clipper (\$99)	\$79
<i>Wallsoft UI Programmer</i> (list \$295)	\$260

OTHER LANGUAGES & TOOLS

<i>Aldebaran Source Print</i> (list \$97).....	\$74	
<i>Aldebaran Tree Diagrammer</i> (list \$77)	\$59	
<i>Digitaltalk Smalltalk/V v2</i> (list \$99)	\$84	<i>New!</i>
<i>Microsoft Fortran w/Codeview</i> (list \$450)	\$270	
<i>Microsoft Quick Basic v3</i> (list \$99)	\$65	
<i>MKS Toolkit Korn shell, vi, grep, 30 more</i> (\$139)	\$115	
<i>Periscope Periscope II</i> (list \$175).....	\$139	
<i>Phoenix Plink-86 +</i> (list \$495).....	\$315	
<i>Sterling Castle Basic Development Tools</i> (\$99)	\$79	
B-Tree & windows w/source for MS & Borland basics		
<i>Texas Instruments PC Scheme</i> great AI into (\$99)	\$77	

**TSF carries hundreds of products
from dozens of publishers. Call or
write for a FREE catalog or a price
quote.**

"How to protect your software by letting people copy it"

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the

protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

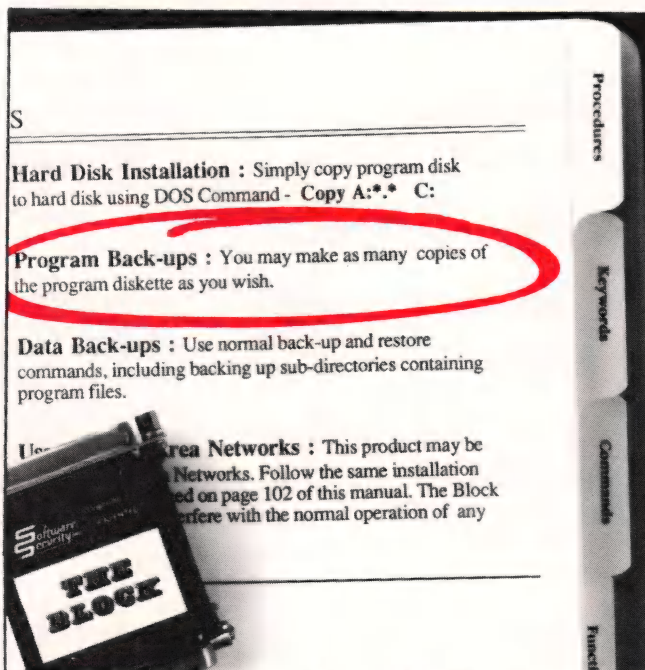
Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment



Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

of the market, or take a stand against the theft of your intellectual property.

"...giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software Security Inc.

870 High Ridge Road Stamford, Connecticut 06905
203 329 8870

RAM-CACHE MANAGER

(continued from page 34)

occupy is to be overwritten. The function specified by the calling routine is called with three parameters:

1. the cache identifier (the fourth parameter to *cacallo()*)
2. the record identifier (generally the block number)
3. a character pointer to the record to process

To implement this, the previous record-writing routine would be changed to the code in Example 3, page 34.

The calling routine must provide an external function to write the record, specified by the initial call to *cacallo()*. When a block marked for processing is needed for some other purpose, the routine is called. An example is shown in Example 4, page 34.

In some applications, a block marked for processing will become obsolete. In this case, the routine *cacuprc()* is called. This routine will keep a block from being sent to the external function that writes the records.

Closing Down the Cache

Because *cacallo()* uses *malloc()* to allocate all the memory the cache uses, it's possible to free that memory with *cacfree()*. All the records that are due for postprocessing are processed at this time by using the *cacflsh()* routine, before the library routine *free()* is called.

Variable Length Records

The cache routines presented in this article are best suited to small- and medium-size caches of large, fixed-length records. Typically, the "records" cached are disk blocks, which contain smaller, logical records. When the records are small, the linear search would become inefficient in short order. Figure 4, right, shows a block diagram of a complete application in which interface programs process requests from the applications.

Conclusion

The routines provided herein can be applied to almost any applica-

tion. They were originally implemented on a Unix System III, 68000-based system and were subsequently ported to other environments. They work well under MS-DOS using all the memory models of the Microsoft C compiler, although a cache of more than 63K does not work.

Availability

All the source code for articles in this issue is available on a single

disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 62.)

Vote for your favorite feature/article.
Circle Reader Service No. 2.

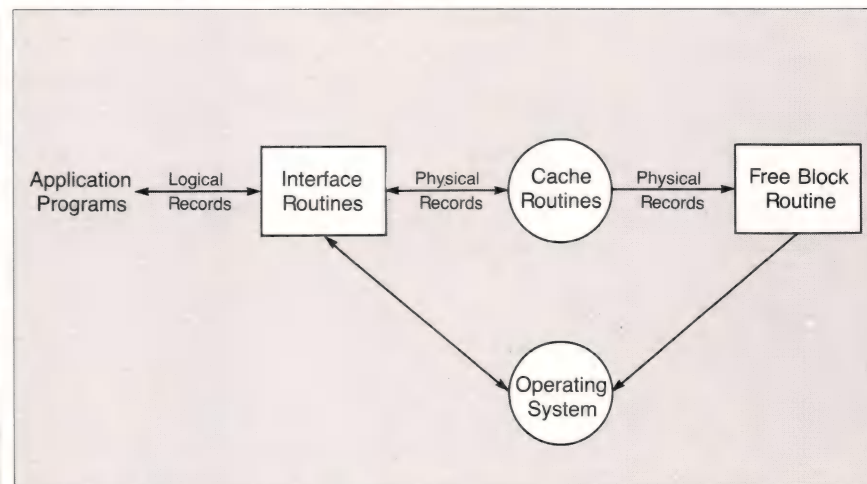


Figure 4: Application interface

UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with CURSES for MS-DOS and MS-OS/2.

THE BETTER PRODUCT. "Aspen Scientific's Curses library is a fine PC version of System V Curses. Screen updating was fast and clean... has good documentation and is available for many compilers...less expensive... its greater flexibility makes it an attractive package for developers." *Computer Language June/87*

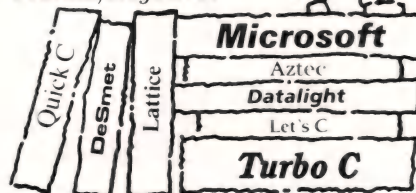
"This is a nice product. If you need Unix-compatible screen output in your programs, or if you just want a nice clean window-management package, I'd recommend it." *Allen Holub, Dr. Dobb's Journal, August/87*

FREE-FAST

UNIX compatible forms tool kit with source code. Included if you **ORDER RIGHT NOW.**

Complete curses tool kit: **\$119.**
Source code available for: \$289.

Look at a few of the applications our Curses is benefiting:
✓ Aerospace
✓ Robotics ✓ Consulting ✓ Telephone
Switching ✓ AI ✓ Voice Recognition
✓ Financial ✓ Engineering ✓ Word Processing



ASPEN SCIENTIFIC

P.O. BOX 72 WHEAT RIDGE,
COLORADO 80034-0072
(303) 423-8088

Trademarks: MS-DOS, MS-Windows, MS-OS/2, XENIX, Quick C (Microsoft); Unix (AT&T Bell Labs); Turbo C (Borland); Aztec (Manx); Lattice (Lattice); Let's C (Mark Williams); DeSmet (DeSmet Software); Datalight (Datalight).

CIRCLE 120 ON READER SERVICE CARD

Putting ROM Code in Its Place

by Rick Naro

Until recently, developers programming for EPROM-based applications didn't have a utility that would take the standard output of MS-DOS tools and spit out a file suitable for burning into EPROM. After spending the last few months longing to use the latest in compilers, such as Microsoft's C or Borland's Turbo C, I decided that perhaps the best tack to take would be to develop my own locate utility.

Although it's an easy matter to put the contents of a .COM file into an EPROM, programs in .EXE format are a different animal altogether. Unlike the binary image of a program found in a .COM format, a .EXE file is a relocatable object module, which requires that the segment references in a program be relocated or adjusted.

Before the .EXE file can be run, a loader must convert the relocatable format into an executable, absolute object module. In the typical MS-DOS system, program loading and segment fix-up is performed by COMMAND.COM and as such is transparent to the user. The COMMAND.COM loader simply reads the relocatable object module into memory, performs the segment fix-up by adjusting the segment references relative to the base segment of the load module, then transfers control to the program.

To support relocation, the .EXE file is partitioned into two compo-

A DOS Locate utility

nents—a header containing the relocation information and the actual binary load module. Both the memory requirements for the program and the initial register values are found in the header, so the first step of loading the binary image is easy.

COMMAND.COM simply requests a suitably sized memory block in which to place the load module, then reads the binary image into that block of memory. Once loaded into memory, segment references are fixed relative to the base segment, using the segment fix-up records. When a programmer codes the following instruction sequence, for example, the assembler and linker cannot determine the final segment value to be used for the *data* segment, and the fix-up is left for the loader to perform:

```
mov ax, data ; Load ax with the data
                                segment
mov ds, ax ; Store in ds
```

Instead, the linker inserts the offset (or virtual segment) of the segment *data* from the base of the load module into the binary object module and inserts an entry in the segment fix-up record pointing to the segment reference requiring fix-up. If the program base is segment 3000h and the virtual segment of

data is 1234h, the loader will perform the fix-up by adding the two and overwriting the segment offset with the sum 4234h (which is the physical segment for the segment *data* in this instance). Once all segment fix-ups have been processed, the loader can transfer control to the new program.

An MS-DOS Locator

For embedded systems, a special type of loader called a locator is required. A loader is distinguished from a locator in two ways: by output destination and by the organization of the absolute object module. Although a loader is designed to write the absolute object module directly to memory (for immediate execution), the output of a locator is an Intel extended hex file suitable for EPROM burning.

Another important feature of a locator is its ability to rearrange segments at arbitrary addresses to reflect the physical organization of the target system. A typical embedded system normally contains EPROM at the upper addresses for the program code and RAM at the lower addresses for data, interrupt vectors, and the stack. Because this organization is incompatible with the contiguous MS-DOS absolute object module, relocating the segments to new addresses is crucial to the operation of the locator. When the locator has finished processing a .EXE file, ROMable code and data will be fixed at addresses in the EPROM address space and volatile data and the stack will be fixed in the RAM address space and the segment fix-ups adjusted to reflect the rearrangement of segments.

Rick Naro is a part time programmer who is interested in software development tools for embedded systems. He can be reached c/o Paradigm Systems, P.O. Box 152, Milford, MA 01757 or as naro on BIX.

By itself, the .EXE file header contains insufficient information for relocation, so the segment map of the program along with instructions on where the segments will be placed in the target system is required. The segment map is prepared by the linker and identifies each segment by name, class, length, and its position within the binary load module. The user must also prepare a configuration file describing the characteristics of the target system and the physical addresses that the program segments will bound. Using both the map and configuration files, the locator can extract and physically relocate the segments to build the ROMable load module.

Although a programmer is normally concerned with segments, they are far too numerous and varied in name to be of much use to the locator. Instead, the locator works with classes. A class is simply a tag applied to a segment by the assembler or compiler. This tag permits the linker to group a segment with other related segments.

For example, each separately compiled source file in the large-memory model will generate a uniquely named code segment, but all such segments will belong to the *code* class. Using the locator directives, a programmer can fix the address of any class and specify the order of a set of classes, configuring the absolute object code for any target hardware.

The locator also needs to process the segment fix-ups but in a slightly different manner from the way the loader does. Each segment listed in the segment map is given an entry in a linked list that contains its segment name, length, virtual segment number, and physical segment number organized by class name. When the configuration file is read, the base segment used for a class is fixed and all physical segment numbers are adjusted by adding this value to the segment offset within the class. Segment fix-up then can proceed with the virtual segment number from the fix-up record used to scan the linked list looking for a match. If found, the corresponding physical segment number is returned and used in the fix-up; otherwise, an unresolved segment error

is reported.

Although the location process sounds simple, there are two pitfalls that must be avoided. As noted previously, an MS-DOS .EXE file is designed to be executed from a contiguous block of memory whereas embedded systems typically have a fragmented address space with pockets of RAM and EPROM placed at the whim of the hardware designer. A potential problem exists in that two adjacent segments in different classes can share a common virtual segment and then be located non-contiguously when the segments are extracted. Because the virtual-to-physical-segment translation in this instance is ambiguous, a situation known as segment aliasing results. Segment aliasing can be avoided by guaranteeing that two segments in different classes never share a common virtual segment. This is easily accomplished by verifying that the first segment in a class is paragraph-aligned or that each segment spans a paragraph boundary.

There is also a potential problem in using groups. A group is a collection of unrelated segments that are organized to fit within, and be addressed as, a single physical segment. Some linkers, such as the Microsoft linker, don't include sufficient information for the locator to reconstruct a group. If groups are used, the user must have information on the organization of the group and include instructions in the configuration file to permit its reconstruction. This is accomplished by using the locate directives to fix the address of the first class in the group and then order the remaining classes in the group as specified by the compiler vendor.

LOCATE

LOCATE is an MS-DOS utility that accepts a relocatable .EXE file and outputs an absolute object module suitable for burning in EPROM. The source code and a make file for building LOCATE can be found in the accompanying listings (beginning on page 68).

Because each application is unique, LOCATE uses several directives to control the location process. These directives are used to identify ROMable classes, assign physical seg-

ments to classes, and specify the order of classes in the absolute load module. Some directives accept a list of one or more operands. The / and \ characters are used whenever an operand is optional and can be repeated zero or more times. Unless otherwise specified, directives and operands are delimited by white space.

The default configuration file has the file name of the input .EXE file with an extension of .CFG. Using a command-line option, the default file name can be overridden and any file can be specified to contain the configuration instructions. This option allows multiple load modules to share a common configuration file.

Class Directive

The *class* directive assigns a physical segment to a class. The first segment in the specified class is assigned the base segment number and the remaining segments in the class are assigned segments relative to the first segment in the class. These segments depend on the length of the preceding segments and the segment alignment.

The *class* directive uses the following syntax:

```
class class = seg
```

where *class* is the name of the class and *seg* is the 16-bit physical segment where the class will be located. For example:

```
class code = 0xfc00
```

assigns the class *code* to segment *fc00h* and therefore the physical address *fc000h*.

Order Directive

The *order* directive is used to specify the ordering of two or more classes. It is important because it allows unrelated classes to be made contiguous without firsthand knowledge of the size and number of segments in the class.

The *order* directive uses the following syntax:

```
order class [class]
```

where the first class in the list was

YOUR SYSTEM'S KEY COMPONENT

The Only Magazine By And For Advanced Micro Users.

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal*. *Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just \$20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!



MICRO/ SYSTEMS JOURNAL

FOR THE
ADVANCED
COMPUTER
USER

SUBSCRIBE
NOW AND

SAVE
OVER
47%

OFF THE
NEWSSTAND
PRICE!



**Yes! I want to subscribe to
Micro/Systems Journal™**

and save over 47% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Savings based on a full one-year cover price of \$47.40. Canada and Mexico add \$6 for surface mail, \$14 for airmail per year. All countries add \$24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC.

2003

47%
SAVINGS



**Yes! I want to subscribe to
Micro/Systems Journal™**
and save over 47% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Savings based on a full one-year cover price of \$47.40. Canada and Mexico add \$6 for surface mail, \$14 for airmail per year. All countries add \$24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC.

2003

47%
SAVINGS



**Yes! I want to subscribe to
Micro/Systems Journal™**
and save over 47% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Savings based on a full one-year cover price of \$47.40. Canada and Mexico add \$6 for surface mail, \$14 for airmail per year. All countries add \$24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC.

2003

47%
SAVINGS



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondido, CA 92025-9843



DOS LOCATE UTILITY
(continued from page 39)

specified in a *class* directive. Any class names specified after the first are located contiguously and aligned to the segment boundary of the first segment in each class. For example:

order code data bss

orders the classes *data* and *bss* immediately following the class *code*.

Dup Directive

The *dup* directive is used to make a copy of the specified class. If used, the *dup* directive should appear before any other directives.

The *dup* directive uses the syntax:

dup class dup_class

where *class* is an existing class and *dup_class* is the name given to the copy of *class*. For example, the directive:

dup data const

makes a copy of the *data* class named *const*. This command is used in conjunction with the *order* directive to locate the *data* and *bss* classes in RAM but force a copy of the class *data* to be included in EPROM for power-on initialization of any initialized data.

If the class *data* contains the initialized data from a compiler, the following commands will locate *data* at address 1000h and create a copy of *data* called *const* to be placed after the *code* class. The start-up code can then initialize the class *data* by copying the *const* class to the *data* class.

dup data const ; Copy the class and
call it const

...
class data = 0x100 ; Fix data at
address 01000h
class code = 0xfc00 ; Fix code at
address fc000h

...
order code const ; Const to immediately follow code
; And read by the startup code

Rom Directive

The *Rom* directive is used to specify

Unmatched.

If you want
unmatched
performance and
portability, we
have it. The
hottest file
handler and
report generator
on the market.

The c-tree file handler offers unmatched file accessing speed. The r-tree report generator makes producing reports a snap. Both packages offer unmatched portability. Thousands of programmers are using these packages in over 50 system environments: DOS, UNIX, XENIX, OS/2, MACINTOSH, VAX, TOWER and YOURS.

More for your money • complete C-source code • single and multi-user capability • no royalties • unlimited free technical support • port to all machines for one price.

c-tree features • fixed and variable length data records • record locking • variable length keys and key compression • overcomes OS file limit and more.

r-tree features • no printer spacing charts • change reports without recoding • unlimited control breaks, accumulators and virtual field calculations • powerful search, select and sort capabilities over multiple files saves days of coding.

FairCom's unmatched products will work for you. Order c-tree today for \$395, r-tree for \$295. When ordered together, r-tree is only \$255. For VISA, MasterCard and C.O.D. orders call (314) 445-6833. For c-tree benchmark comparisons, write us at 4006 West Broadway, Columbia, MO 65203.

CIRCLE 121 ON READER SERVICE CARD



c-tree / r-tree
By FairCom

4006 W. Broadway Columbia, MO 65203

GET THE BASIC FACTS

Introducing ApBasic™, the only program available today that offers a **real** debugger feature, with single-step, watch variables and breakpoints.

Plus, on-line help for language, one megabyte of code space and string space, a true native code compiler (no psuedo-code) and structured code. And, because it's ApBasic, there's more!

Editor Features • Fast full screen editor with undelete lines, block copy and move, search and replace, one keystroke to compile and run • **Language Features** • 8087/80287 support • Alphanumeric line labels, no more line numbers • Multi-line if statements • Supports DOS 3.xx networks • Named constants • Text windows • Modular sub-programs and multi-line functions with local, static and global variables up to 64K of code each • **Debugger Features** • Up to twenty breakpoints • Up to ten watch variables • View output screen while in debugger • single-step or go to line with cursor • **System Features** • Source code toolboxes available soon • Creates stand-alone .EXE files or smaller chain files • Compact compiler, runs easily on a single diskette computer.

INTRODUCTORY OFFER

\$69.95

OFFER EXPIRES DEC. 31, 1987

When you get the facts,
there is a basic difference.

ApBasic
BY COMTECH

1 · 904 · 497 · 4810

Comtech Software and Consulting, Inc.
P.O. Box 280, Ft. White, FL 30238
TELEX 910 · 250 · 4832 COMTECH UQ

DOS LOCATE UTILITY (continued from page 41)

which classes are ROMable. Classes containing program code and constant data need to be located and placed in ROM to be available when

the system is powered up and initialized. Other classes such as uninitialized data and the stack require only to be located at a physical address and do not need to be placed in the output file.

The *rom* directive uses the follow-

```
/* This program demonstrates the use of the LOCATE utility. It
contains all of the components of a typical C program to
exercise the startup code and locate utility.

*/
char *ptr = "class DATA" ; /* Initialized data */
int array[10][10] ; /* Uninitialized data */

main()
{
    int i, j ; /* Automatics */
    static char *s = "" ; /* Static initialized data */

    for (i = 0; i < 10; i++) {
        for (j = 0; j < 10; j++)
            array[i][j] = i * j ;
    }

    strcpy(s, ptr) ; /* Bring in a library function */
}
```

Example 1: The demonstration program

```
C>masm /MX tc, tc, tc ;
Microsoft (R) Macro Assembler Version 4.00
Copyright (C) Microsoft Corp 1981, 1983, 1984, 1985.
All rights reserved.
49272 Bytes symbol space free
0 Warning Errors
0 Severe Errors

C>tcc -c -ml demo
Turbo C Version 1.0 Copyright (c) 1987 Borland International
demo.c:

Available memory 293342
```

Example 2: Compiling the C source code and the MASM start-up module

```
C>type demo.map

Start Stop Length Name Class
00000H 00035H 00036H _TEXT CODE
00036H 0007FH 0004AH DEMO_TEXT CODE
00080H 000A8H 00029H STRCPY_TEXT CODE
000B0H 000BFH 00010H _ETEXT CODEEND
000C0H 000D3H 00014H _DATA DATA
000E0H 001A7H 000C8H _BSS BSS
001A8H 001A8H 00000H _BSEND BSSEND
001B0H 003AFH 00200H _STACK STACK

Address Publics by Value
0000:0000 START
0003:0006 _MAIN
0008:0000 _STRCPY
000B:0010 TEND
000C:0000 PTR
000C:0000 IDATA
000C:0020 ARRAY
000C:0020 BDATA
000C:00E8 EDATA
001B:0200 TOS

Program entry point at 0000:0000
```

Example 3: The linker map file

slipping away.

SOURCE CODE

CREATED SCREEN



Trademark owners: IBM C and IBM BASIC are trademarks of International Business Machines Corp.; dBASE is a product of Ashton-Tate Corp.; Quick BASIC is a trademark of Microsoft Inc.; Turbo BASIC, Turbo Pascal, Turbo C are trademarks of Borland International Corp.
© 1986, 1987 SoftScience Corporation.

CIRCLE 123 ON READER SERVICE CARD

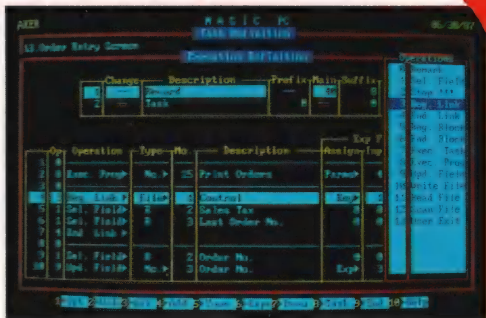
MAGIC PC: A REVOLUTION IN POWER, PRICE & PROGRAMMING SPEED.

You know how database applications are created — by hacking out line after line of time-consuming code. Most DBMS' and 4GL's give you some programming power. But when it comes to serious applications, they keep you bolted to your seat writing mountains of tedious code. And rewriting it all over again with every design change.

Imagine how much faster you'd be if you could replace the painful coding phase with an innovative visual technology which takes only a fraction of the time: Introducing Magic PC—the revolutionary Visual Database Language from Aker Corporation:

High-Speed Programming:

With Magic PC's visual design language you quickly describe your programs in non-procedural Execution Tables. They contain compact programming operations which are executed by Magic PC's runtime engine. You fill-in the tables using a visual interface driven by windows and point-and-shoot menus. One table with 50 operations eliminates writing more than 500 traditional lines of code. Yet with Magic PC you don't sacrifice any power or flexibility.



With a powerful set of high-level non-procedural operations you program at only a fraction of the time.

Maximum Power And Simplicity:

With Magic PC, you can generate robust DBMS applications including screens, windows, menus, reports, forms, import/export, and much more! Plus, Magic PC has one of the friendliest user interfaces you've ever seen. Using Magic PC you can look-up and transfer data through a powerful Zoom Window system. Magic PC even lets you perform command-free queries.

Btrieve Performance:

Magic PC incorporates Btrieve, the high-performance file manager from SoftCraft. This gives you exceptional access speed, extended data dictionary capabilities, and automatic file recovery!

Virtually Maintenance-Free:

With Magic PC you can modify your application design "on the fly" without any manual maintenance. Magic PC automatically updates your programs and data files on-line! This also makes Magic PC an ideal tool for prototyping complete applications in hours instead of days.

FREE Networking:

Magic PC comes complete with LAN features. Develop multi-user applications for your LAN with Magic's file and record-locking security levels.

Stand-Alone Runtime:

Distribute your applications and protect your design with Magic PC's low cost runtime engine.

All For Only \$199:

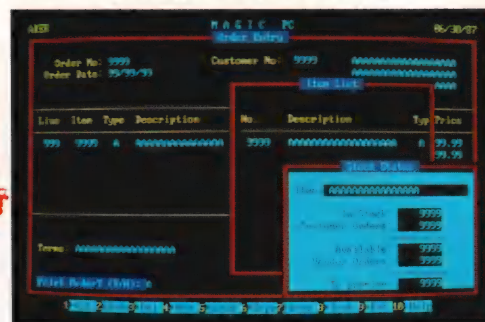
Best of all, Magic PC is an unbeatable bargain. For a limited time, Magic PC's price has been reduced to only \$199! Yes, this is the same Magic PC that normally lists for \$695! And Magic PC eliminates the need for a separate DBMS, compiler, or application generator. It comes complete with all the tools you need to develop your own database applications instantly.

CIRCLE 124 ON READER SERVICE CARD

\$199 - With A Money-Back Guarantee!

For a limited time, you can get Magic PC for only \$199. And even at this low price, Magic PC is risk-free. If you're not completely satisfied, simply return it within 30 days and we'll buy it back (less \$19.95 restocking fee). And if you'd like a preview, Magic PC's Tutorial Demo is available for just \$19.95.

But you'd better hurry — Magic PC's special \$199 price won't last long!



Pop-up Zoom Windows run multiple programs per screen — with point-and-shoot data transfer between windows!

Join The Magic PC Revolution

To unleash your DBMS design power, order your \$199 copy of Magic PC right now by calling toll-free or returning the coupon below.

ORDER NOW: CALL
(800) 345-MAGIC
In CA (714) 250-1718

"Magic PC's data base engine delivers powerful applications in a fraction of the time... there is truly no competitive product."

Victor Wright — PC Tech Journal

Also recommended by: PC Magazine, PC World, PC Week, Computer Language, Data Base Advisor, and many other publications worldwide.

MAGIC PC
The Visual Database Language

by **AKER**

Yes! I want to generate powerful applications much faster!

☐ Rush me my copy of Magic PC at the special promotional price of \$199 (add \$10 P&H, and tax in CA. International orders add \$30). I understand I can return Magic PC for a refund within 30 days, if I'm not completely satisfied.*

☐ Rush me a copy of Magic PC Tutorial Demo at \$19.95 (add \$5 P&H, and tax in CA. International orders add \$15).

Name _____

Company _____

Street Address (no POB) _____

City _____ State _____ Zip _____

☐ Check enclosed ☐ Charge to my: ☐ VISA ☐ MC ☐ AMERICAN EXPRESS

Account No. _____

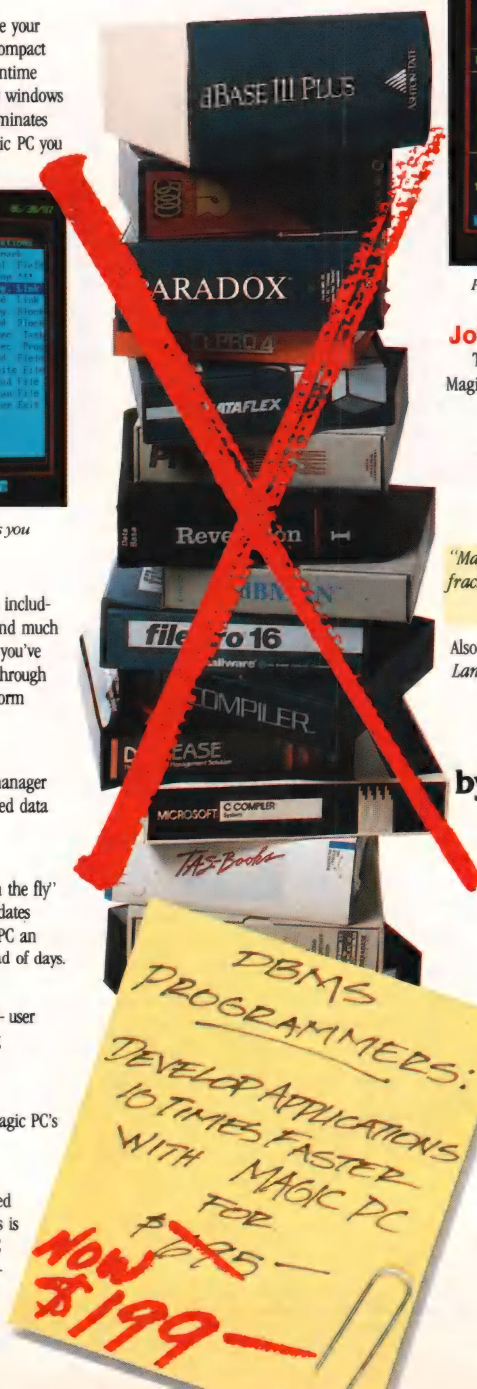
Acct. Name _____ Exp. Date _____

Signature _____

Return to: **Aker Corp., 18007 Skyport Cir B2, Irvine, CA 92714**

System requirements: IBM PC, XT, AT, PS/2 or 100% compatible with 512K RAM, hard disk and DOS 2.0 or later. 5¼" format, not copy protected. Dealer pricing available. *Return policy valid in US only.

Aker, Magic PC, The Visual Database Language are trademarks of Aker Corporation. All other trademarks acknowledged. © Copyright 1987, Aker Corp.



DOS LOCATE UTILITY

(continued from page 42)

ing syntax:

rom class [class]

For example:

rom code const

forces the classes *code* and *const* to be placed in the output object file.

Comments

To aid in documenting the location process, comments can be added to the tail of any command line or as a separate line. Comments begin with a semicolon (;) and continue to the end of the line. Blank lines and comments can appear freely within the configuration file for documentation and readability.

Options

In order to provide a degree of flexibility, the LOCATE utility can accept command-line options (or switches if you prefer) that influence the operation of the locator. Command-line options are lowercase letters introduced with a leading dash (-) with no white space between the option letter and the argument. Some examples of LOCATE command lines are:

locate -b hello

locate -b -ccommon.cfg hello

locate -hello.hx -b hello

A command line begins with *locate*, is followed by zero or more options, and is terminated with the path name of the file to be located. In the following descriptions, left and right brackets ([and]) are used to denote mandatory arguments.

-b—The default setting for LOCATE is to generate an Intel extended hex start address record containing the entry point of the program. By specifying the **-b** option, LOCATE will create an absolute segment at address *ffff:0* and place an intersegment jump instruction to the entry point of the program.

-c[filename]—specifies a different configuration file. The default configuration file is *filename .CFG*,

where *filename* is the name of the load module. One use of this option is to allow different object modules to be located using a shared configuration file.

-h[filename]—changes the name of the Intel extended hex output file. Normally, the output is placed in a file with the same name as the .EXE input file and a default extension of .HEX.

-p[filename]—changes the name of the locate map file containing the segment assignments and public

symbols. Normally, the locate statistics are placed in a file with the same name as the .EXE input file and a default extension of .LOC.

LOCATE Example

To demonstrate the use of LOCATE, I'll now discuss an example that uses the Turbo C compiler from Borland. Using the large-memory model, the program in Example 1, page 42, loads several different code and data segments that can then be processed by LOCATE. The com-

```
;
; This configuration file is used with Turbo C to build a
; ROMable image. It defines physical addresses for three
; classes, makes a copy of the initialized data class to
; keep in ROM and instructs the locator the order of the
; different classes.

dup      DATA CONST          ; Make a copy of the initialized data class
                                ; and name it CONST

class CODE = 0xfc00           ; Start code at address FC000H
class STACK = 0x0080          ; The stack at address 00800H
class DATA = 0x0100          ; DGROUP at address 01000H
order DATA BSS BSEND         ; Define the order of DGROUP
order CODE CODEEND CONST      ; And the order of classes in ROM

rom CODE CONST                ; ROM only the program code and the copy
                                ; of the initialized data that the startup
                                ; code copies from ROM to DGROUP
```

Example 4: The configuration file

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2**. C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: \$195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.

Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for \$195:

- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:

send check or money order to:

GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563

To order with Visa or MC,
phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

DOS LOCATE UTILITY

(continued from page 45)

piled C source and Turbo C run-time routines are linked with the Turbo C assembly-language start-up code, TC.ASM. The start-up code together with the *order* and *dup* directives demonstrates how the locator can initialize the *data* class and zero out the *bss* class.

As shown in Example 2, page 42, you begin by compiling the C source and the MASM start-up module. The Turbo C options used are to disable linking (*-c*) and select the large-memory model.

I disable the automatic link following the compile so that I can substitute a ROMable version of the C start-up. The Turbo C large-memory-model library is searched to satisfy the external reference to the *strcpy()* function, as follows:

```
C>link /m tc demo, demo, demo,
                        \turboc\lib\cl
Turbo Link Version 1.0 Copyright (c)
1987 Borland International
```

For reference, the linker map file for this example is reproduced in Example 3, page 42. Note the segment and class assignments and watch how the locator processes and converts the executable image to a ROM-

able image.

The configuration file for the example (see Example 4, page 45) must be able to handle the group and the initialized data generated by Turbo C. LOCATE is instructed to make a copy of the *data* class that contains the program initialized data. Next the base segments of the three independent classes (*code*, *data*, and *stack*) are specified using the *class* directive. The *order* directive is used to recreate the Turbo C *dgroup* and fix the copy of the initialized data segment immediately following the *codeend* class. With it tucked nicely in EPROM and its physical address determined by the *tend* label, the start-up code can copy the class *const* to the *data* class before calling *main()*.

LOCATE is then executed to process the .EXE file and output the absolute load module, as follows:

```
C>locate -b demo
MS-DOS Locate Utility
Copyright (C) 1987 Paradigm Systems. All rights reserved
```

The output from the locator, shown in Example 5, page 46, is an Intel extended hex file and a segment map detailing the new physical address assignments. The locate map also contains a list of public sym-

bols for use in debugging the target system. Note how the segments and classes have been relocated according to the instructions in the configuration file and correspond to the addresses of the target hardware.

The file DEMO.HEX (in Example 6, page 46) is now ready to be sent to the EPROM programmer. In an 8-bit system, the data is burned directly into one or more EPROMs. In a 16-bit bus system, the EPROM programmer must be used to split the load module into upper and lower bytes for programming the upper and lower bytes in separate EPROMs.

Summary

Although a simple example, the sample program demonstrates the power and flexibility of turning a low-cost PC into a powerful, embedded, system development tool for the NEC and Intel microprocessors. With access to a wide range of popular software development tools, program development for embedded systems has never been easier.

DDJ

(Listings begin on page 68.)

Vote for your favorite feature/article.
Circle Reader Service No. 3.

```
C>type demo.loc
MS-DOS Locate Utility Version 1.0

Input File: DEMO.EXE
Output File: DEMO.HEX
Configuration File: DEMO.CFG
Invoked by: C:\BIN\LOCATE.EXE -b demo
Date/Time: Mon Aug 03 14:40:17 1987
```

Segment Information

Name	Class	Address	Length
TEXT	CODE	FC000H	0036H
DEMO_TEXT	CODE	FC036H	004AH
STRCPY_TEX	CODE	FC080H	0029H
_ETEXT	CODEEND	FC0B0H	0010H
_DATA	DATA	01000H	0014H
_BSS	BSS	01020H	00C8H
_BSEND	BSEND	010E8H	0000H
_STACK	STACK	00800H	0200H
_DATA	NEWDATA	FC0C0H	0014H
??BOOT	(ABSOLUTE)	FFFF0H	0005H

Public Symbols

FC00:0000	START	FC03:0006	MAIN
FC08:0000	STRCPY	FC0B:0010	TEND
0100:00E8	EDATA	0100:0020	BADATA
0100:0020	_ARRAY	0100:0000	IDATA
0100:0000	_PTR	0080:0200	TOS

Entry Point - FC00:0000

```
:02000002FC0000
:10000000FAB880008ED0BC0002B800018EC0B80BD8
:10001000FC408ED8BE00008BFEB920002BCFF3A48D
:10002000061F32C0BF2000B9E8002BCFF3AAFB9A0D
:06003000060003FCEBCA10
:02000002FC03FD
:10000600565733F6EB2433FFEB1A8BC6F7E7508BC4
:10001600C6BA1400F7E28BD88BC7D1E003D858894B
:100026008720004783FF0A7CE14683FE0A7CD7FFD0
:10003600360200FF360000FF360600FF3604009A3F
:0A0046000000008FC83C4085F5ECBD5
:02000002FC08F8
:100000005657558BECFC1EC47E0E8BF732C0B9FFE1
:10001000FFF2AEF7D18CC38EDBC47E0AF3A41F8B34
:09002000560C8B460A5D5F5ECBB5
:02000002FC0CF4
:100000008000000113000001436C61737320444138
:040010005441000057
:02000002FFFFFFE
:05000000EA000000FC15
:04000003FC000000FD
:00000001FF
```

Example 5: The output from the locator

Example 6: The file DEMO.HEX

Clarify and document your source listing and get an "organization chart" of your program's structure with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE®, FORTRAN and Modula-2 programmers.

Now works with FORTRAN

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine
Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only \$155.00.**

800-257-5773 Dept. 58
In California:
800-257-5774 Dept. 58

MasterCard, VISA, American Express, COD. Add \$5 for shipping/handling.

or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

Source Print™

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.

\$97⁰⁰

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Before

```
1 source ()
2 while (lar < nres && ares[lar][0] == 0)
3 {
4   if ((d = ares[lar][1]) == 0)
5   {
6     p = a(ares[lar][1]);
7     while (d = p)
8     {
9       loop++;
10      loop++;
11    }
12    lar++;
13  }
14 }
15
```

Before

After

Wed 12-31-86 07:22:03 INDEX (Cross Ref)
all identifiers

inrecord	4.191	9=396	19.825	19=826
	21.889	22.922	22.953	23=978
	23.990			
ina	53.2293	53=2309	53=2319	53.2325
	54.2331	54.2332	54.2336	54=2346
	54.2354	54.2364	54.2365	54.2366
intext	4.193	9=395	43.1796	43.1815
	43=1820	45=1902		

Index

04-06-86 13:45:44 dms7.prg
Sun 04-06-86 13:47:57

```
1 PUBLIC value, val1, val2, val3
2 USE val1val2 index dates
3 date1="0000" "12/31/85"
4 DO WHILE date1 < "0001/01/86"
5   date1 = date1 + 1
6   #1:10 say "Enter date1:" get da
7   #2:10 say "Enter date2:" get da
8   value = 0.00
9   val1 = 0.00
10  val2 = 0.00
11  val3 = 0.00
12  IF TRIM(date1) = "0000"
13  THEN
14    CASE selector = "1"
15    DO proc1
16    CASE selector = "2"
17    DO proc2
18    IF value = 0.00
19    THEN
20      value = value + 1
21      val1 = val1 + 1
22      val2 = val2 + 1
23      val3 = val3 + 1
24    CASE selector = "3"
25    IF value = 0.00
26    THEN
27      value = value + 1
28      val1 = val1 + 1
29      val2 = val2 + 1
30      val3 = val3 + 1
31    CASE selector = "4"
32    val1 = val1 + 1
33    val2 = val2 + 1
34    val3 = val3 + 1
35    val1 = val1 + 1
36    val2 = val2 + 1
37    val3 = val3 + 1
38    val1 = val1 + 1
39    val2 = val2 + 1
40    val3 = val3 + 1
41    val1 = val1 + 1
42    val2 = val2 + 1
43    val3 = val3 + 1
44    val1 = val1 + 1
45    val2 = val2 + 1
46    val3 = val3 + 1
47    val1 = val1 + 1
48    val2 = val2 + 1
49    val3 = val3 + 1
50    val1 = val1 + 1
51    val2 = val2 + 1
52    val3 = val3 + 1
53    val1 = val1 + 1
54    val2 = val2 + 1
55    val3 = val3 + 1
56    val1 = val1 + 1
57    val2 = val2 + 1
58    val3 = val3 + 1
59    val1 = val1 + 1
60    val2 = val2 + 1
61    val3 = val3 + 1
62    val1 = val1 + 1
63    val2 = val2 + 1
64    val3 = val3 + 1
65    val1 = val1 + 1
66    val2 = val2 + 1
67    val3 = val3 + 1
68    val1 = val1 + 1
69    val2 = val2 + 1
70    val3 = val3 + 1
71    val1 = val1 + 1
72    val2 = val2 + 1
73    val3 = val3 + 1
74    val1 = val1 + 1
75    val2 = val2 + 1
76    val3 = val3 + 1
77    val1 = val1 + 1
78    val2 = val2 + 1
79    val3 = val3 + 1
80    val1 = val1 + 1
81    val2 = val2 + 1
82    val3 = val3 + 1
83    val1 = val1 + 1
84    val2 = val2 + 1
85    val3 = val3 + 1
86    val1 = val1 + 1
87    val2 = val2 + 1
88    val3 = val3 + 1
89    val1 = val1 + 1
90    val2 = val2 + 1
91    val3 = val3 + 1
92    val1 = val1 + 1
93    val2 = val2 + 1
94    val3 = val3 + 1
95    val1 = val1 + 1
96    val2 = val2 + 1
97    val3 = val3 + 1
98    val1 = val1 + 1
99    val2 = val2 + 1
100   val3 = val3 + 1
101   val1 = val1 + 1
102   val2 = val2 + 1
103   val3 = val3 + 1
104   val1 = val1 + 1
105   val2 = val2 + 1
106   val3 = val3 + 1
107   val1 = val1 + 1
108   val2 = val2 + 1
109   val3 = val3 + 1
110   val1 = val1 + 1
111   val2 = val2 + 1
112   val3 = val3 + 1
113   val1 = val1 + 1
114   val2 = val2 + 1
115   val3 = val3 + 1
116   val1 = val1 + 1
117   val2 = val2 + 1
118   val3 = val3 + 1
119   val1 = val1 + 1
120   val2 = val2 + 1
121   val3 = val3 + 1
122   val1 = val1 + 1
123   val2 = val2 + 1
124   val3 = val3 + 1
125   val1 = val1 + 1
126   val2 = val2 + 1
127   val3 = val3 + 1
128   val1 = val1 + 1
129   val2 = val2 + 1
130   val3 = val3 + 1
131   val1 = val1 + 1
132   val2 = val2 + 1
133   val3 = val3 + 1
134   val1 = val1 + 1
135   val2 = val2 + 1
136   val3 = val3 + 1
137   val1 = val1 + 1
138   val2 = val2 + 1
139   val3 = val3 + 1
140   val1 = val1 + 1
141   val2 = val2 + 1
142   val3 = val3 + 1
143   val1 = val1 + 1
144   val2 = val2 + 1
145   val3 = val3 + 1
146   val1 = val1 + 1
147   val2 = val2 + 1
148   val3 = val3 + 1
149   val1 = val1 + 1
150   val2 = val2 + 1
151   val3 = val3 + 1
152   val1 = val1 + 1
153   val2 = val2 + 1
154   val3 = val3 + 1
155   val1 = val1 + 1
156   val2 = val2 + 1
157   val3 = val3 + 1
158   val1 = val1 + 1
159   val2 = val2 + 1
160   val3 = val3 + 1
161   val1 = val1 + 1
162   val2 = val2 + 1
163   val3 = val3 + 1
164   val1 = val1 + 1
165   val2 = val2 + 1
166   val3 = val3 + 1
167   val1 = val1 + 1
168   val2 = val2 + 1
169   val3 = val3 + 1
170   val1 = val1 + 1
171   val2 = val2 + 1
172   val3 = val3 + 1
173   val1 = val1 + 1
174   val2 = val2 + 1
175   val3 = val3 + 1
176   val1 = val1 + 1
177   val2 = val2 + 1
178   val3 = val3 + 1
179   val1 = val1 + 1
180   val2 = val2 + 1
181   val3 = val3 + 1
182   val1 = val1 + 1
183   val2 = val2 + 1
184   val3 = val3 + 1
185   val1 = val1 + 1
186   val2 = val2 + 1
187   val3 = val3 + 1
188   val1 = val1 + 1
189   val2 = val2 + 1
190   val3 = val3 + 1
191   val1 = val1 + 1
192   val2 = val2 + 1
193   val3 = val3 + 1
194   val1 = val1 + 1
195   val2 = val2 + 1
196   val3 = val3 + 1
197   val1 = val1 + 1
198   val2 = val2 + 1
199   val3 = val3 + 1
200   val1 = val1 + 1
201   val2 = val2 + 1
202   val3 = val3 + 1
203   val1 = val1 + 1
204   val2 = val2 + 1
205   val3 = val3 + 1
206   val1 = val1 + 1
207   val2 = val2 + 1
208   val3 = val3 + 1
209   val1 = val1 + 1
210   val2 = val2 + 1
211   val3 = val3 + 1
212   val1 = val1 + 1
213   val2 = val2 + 1
214   val3 = val3 + 1
215   val1 = val1 + 1
216   val2 = val2 + 1
217   val3 = val3 + 1
218   val1 = val1 + 1
219   val2 = val2 + 1
220   val3 = val3 + 1
221   val1 = val1 + 1
222   val2 = val2 + 1
223   val3 = val3 + 1
224   val1 = val1 + 1
225   val2 = val2 + 1
226   val3 = val3 + 1
227   val1 = val1 + 1
228   val2 = val2 + 1
229   val3 = val3 + 1
230   val1 = val1 + 1
231   val2 = val2 + 1
232   val3 = val3 + 1
233   val1 = val1 + 1
234   val2 = val2 + 1
235   val3 = val3 + 1
236   val1 = val1 + 1
237   val2 = val2 + 1
238   val3 = val3 + 1
239   val1 = val1 + 1
240   val2 = val2 + 1
241   val3 = val3 + 1
242   val1 = val1 + 1
243   val2 = val2 + 1
244   val3 = val3 + 1
245   val1 = val1 + 1
246   val2 = val2 + 1
247   val3 = val3 + 1
248   val1 = val1 + 1
249   val2 = val2 + 1
250   val3 = val3 + 1
251   val1 = val1 + 1
252   val2 = val2 + 1
253   val3 = val3 + 1
254   val1 = val1 + 1
255   val2 = val2 + 1
256   val3 = val3 + 1
257   val1 = val1 + 1
258   val2 = val2 + 1
259   val3 = val3 + 1
260   val1 = val1 + 1
261   val2 = val2 + 1
262   val3 = val3 + 1
263   val1 = val1 + 1
264   val2 = val2 + 1
265   val3 = val3 + 1
266   val1 = val1 + 1
267   val2 = val2 + 1
268   val3 = val3 + 1
269   val1 = val1 + 1
270   val2 = val2 + 1
271   val3 = val3 + 1
272   val1 = val1 + 1
273   val2 = val2 + 1
274   val3 = val3 + 1
275   val1 = val1 + 1
276   val2 = val2 + 1
277   val3 = val3 + 1
278   val1 = val1 + 1
279   val2 = val2 + 1
280   val3 = val3 + 1
281   val1 = val1 + 1
282   val2 = val2 + 1
283   val3 = val3 + 1
284   val1 = val1 + 1
285   val2 = val2 + 1
286   val3 = val3 + 1
287   val1 = val1 + 1
288   val2 = val2 + 1
289   val3 = val3 + 1
290   val1 = val1 + 1
291   val2 = val2 + 1
292   val3 = val3 + 1
293   val1 = val1 + 1
294   val2 = val2 + 1
295   val3 = val3 + 1
296   val1 = val1 + 1
297   val2 = val2 + 1
298   val3 = val3 + 1
299   val1 = val1 + 1
300   val2 = val2 + 1
301   val3 = val3 + 1
302   val1 = val1 + 1
303   val2 = val2 + 1
304   val3 = val3 + 1
305   val1 = val1 + 1
306   val2 = val2 + 1
307   val3 = val3 + 1
308   val1 = val1 + 1
309   val2 = val2 + 1
310   val3 = val3 + 1
311   val1 = val1 + 1
312   val2 = val2 + 1
313   val3 = val3 + 1
314   val1 = val1 + 1
315   val2 = val2 + 1
316   val3 = val3 + 1
317   val1 = val1 + 1
318   val2 = val2 + 1
319   val3 = val3 + 1
320   val1 = val1 + 1
321   val2 = val2 + 1
322   val3 = val3 + 1
323   val1 = val1 + 1
324   val2 = val2 + 1
325   val3 = val3 + 1
326   val1 = val1 + 1
327   val2 = val2 + 1
328   val3 = val3 + 1
329   val1 = val1 + 1
330   val2 = val2 + 1
331   val3 = val3 + 1
332   val1 = val1 + 1
333   val2 = val2 + 1
334   val3 = val3 + 1
335   val1 = val1 + 1
336   val2 = val2 + 1
337   val3 = val3 + 1
338   val1 = val1 + 1
339   val2 = val2 + 1
340   val3 = val3 + 1
341   val1 = val1 + 1
342   val2 = val2 + 1
343   val3 = val3 + 1
344   val1 = val1 + 1
345   val2 = val2 + 1
346   val3 = val3 + 1
347   val1 = val1 + 1
348   val2 = val2 + 1
349   val3 = val3 + 1
350   val1 = val1 + 1
351   val2 = val2 + 1
352   val3 = val3 + 1
353   val1 = val1 + 1
354   val2 = val2 + 1
355   val3 = val3 + 1
356   val1 = val1 + 1
357   val2 = val2 + 1
358   val3 = val3 + 1
359   val1 = val1 + 1
360   val2 = val2 + 1
361   val3 = val3 + 1
362   val1 = val1 + 1
363   val2 = val2 + 1
364   val3 = val3 + 1
365   val1 = val1 + 1
366   val2 = val2 + 1
367   val3 = val3 + 1
368   val1 = val1 + 1
369   val2 = val2 + 1
370   val3 = val3 + 1
371   val1 = val1 + 1
372   val2 = val2 + 1
373   val3 = val3 + 1
374   val1 = val1 + 1
375   val2 = val2 + 1
376   val3 = val3 + 1
377   val1 = val1 + 1
378   val2 = val2 + 1
379   val3 = val3 + 1
380   val1 = val1 + 1
381   val2 = val2 + 1
382   val3 = val3 + 1
383   val1 = val1 + 1
384   val2 = val2 + 1
385   val3 = val3 + 1
386   val1 = val1 + 1
387   val2 = val2 + 1
388   val3 = val3 + 1
389   val1 = val1 + 1
390   val2 = val2 + 1
391   val3 = val3 + 1
392   val1 = val1 + 1
393   val2 = val2 + 1
394   val3 = val3 + 1
395   val1 = val1 + 1
396   val2 = val2 + 1
397   val3 = val3 + 1
398   val1 = val1 + 1
399   val2 = val2 + 1
400   val3 = val3 + 1
401   val1 = val1 + 1
402   val2 = val2 + 1
403   val3 = val3 + 1
404   val1 = val1 + 1
405   val2 = val2 + 1
406   val3 = val3 + 1
407   val1 = val1 + 1
408   val2 = val2 + 1
409   val3 = val3 + 1
410   val1 = val1 + 1
411   val2 = val2 + 1
412   val3 = val3 + 1
413   val1 = val1 + 1
414   val2 = val2 + 1
415   val3 = val3 + 1
416   val1 = val1 + 1
417   val2 = val2 + 1
418   val3 = val3 + 1
419   val1 = val1 + 1
420   val2 = val2 + 1
421   val3 = val3 + 1
422   val1 = val1 + 1
423   val2 = val2 + 1
424   val3 = val3 + 1
425   val1 = val1 + 1
426   val2 = val2 + 1
427   val3 = val3 + 1
428   val1 = val1 + 1
429   val2 = val2 + 1
430   val3 = val3 + 1
431   val1 = val1 + 1
432   val2 = val2 + 1
433   val3 = val3 + 1
434   val1 = val1 + 1
435   val2 = val2 + 1
436   val3 = val3 + 1
437   val1 = val1 + 1
438   val2 = val2 + 1
439   val3 = val3 + 1
440   val1 = val1 + 1
441   val2 = val2 + 1
442   val3 = val3 + 1
443   val1 = val1 + 1
444   val2 = val2 + 1
445   val3 = val3 + 1
446   val1 = val1 + 1
447   val2 = val2 + 1
448   val3 = val3 + 1
449   val1 = val1 + 1
450   val2 = val2 + 1
451   val3 = val3 + 1
452   val1 = val1 + 1
453   val2 = val2 + 1
454   val3 = val3 + 1
455   val1 = val1 + 1
456   val2 = val2 + 1
457   val3 = val3 + 1
458   val1 = val1 + 1
459   val2 = val2 + 1
460   val3 = val3 + 1
461   val1 = val1 + 1
462   val2 = val2 + 1
463   val3 = val3 + 1
464   val1 = val1 + 1
465   val2 = val2 + 1
466   val3 = val3 + 1
467   val1 = val1 + 1
468   val2 = val2 + 1
469   val3 = val3 + 1
470   val1 = val1 + 1
471   val2 = val2 + 1
472   val3 = val3 + 1
473   val1 = val1 + 1
474   val2 = val2 + 1
475   val3 = val3 + 1
476   val1 = val1 + 1
477   val2 = val2 + 1
478   val3 = val3 + 1
479   val1 = val1 + 1
480   val2 = val2 + 1
481   val3 = val3 + 1
482   val1 = val1 + 1
483   val2 = val2 + 1
484   val3 = val3 + 1
485   val1 = val1 + 1
486   val2 = val2 + 1
487   val3 = val3 + 1
488   val1 = val1 + 1
489   val2 = val2 + 1
490   val3 = val3 + 1
491   val1 = val1 + 1
492   val2 = val2 + 1
493   val3 = val3 + 1
494   val1 = val1 + 1
495   val2 = val2 + 1
496   val3 = val3 + 1
497   val1 = val1 + 1
498   val2 = val2 + 1
499   val3 = val3 + 1
500   val1 = val1 + 1
501   val2 = val2 + 1
502   val3 = val3 + 1
503   val1 = val1 + 1
504   val2 = val2 + 1
505   val3 = val3 + 1
506   val1 = val1 + 1
507   val2 = val2 + 1
508   val3 = val3 + 1
509   val1 = val1 + 1
510   val2 = val2 + 1
511   val3 = val3 + 1
512   val1 = val1 + 1
513   val2 = val2 + 1
514   val3 = val3 + 1
515   val1 = val1 + 1
516   val2 = val2 + 1
517   val3 = val3 + 1
518   val1 = val1 + 1
519   val2 = val2 + 1
520   val3 = val3 + 1
521   val1 = val1 + 1
522   val2 = val2 + 1
523   val3 = val3 + 1
524   val1 = val1 + 1
525   val2 = val2 + 1
526   val3 = val3 + 1
527   val1 = val1 + 1
528   val2 = val2 + 1
529   val3 = val3 + 1
530   val1 = val1 + 1
531   val2 = val2 + 1
532   val3 = val3 + 1
533   val1 = val1 + 1
534   val2 = val2 + 1
535   val3 = val3 + 1
536   val1 = val1 + 1
537   val2 = val2 + 1
538   val3 = val3 + 1
539   val1 = val1 + 1
540   val2 = val2 + 1
541   val3 = val3 + 1
542   val1 = val1 + 1
543   val2 = val2 + 1
544   val3 = val3 + 1
545   val1 = val1 + 1
546   val2 = val2 + 1
547   val3 = val3 + 1
548   val1 = val1 + 1
549   val2 = val2 + 1
550   val3 = val3 + 1
551   val1 = val1 + 1
552   val2 = val2 + 1
553   val3 = val3 + 1
554   val1 = val1 + 1
555   val2 = val2 + 1
556   val3 = val3 + 1
557   val1 = val1 + 1
558   val2 = val2 + 1
559   val3 = val3 + 1
560   val1 = val1 + 1
561   val2 = val2 + 1
562   val3 = val3 + 1
563   val1 = val1 + 1
564   val2 = val2 + 1
565   val3 = val3 + 1
566   val1 = val1 + 1
567   val2 = val2 + 1
568   val3 = val3 + 1
569   val1 = val1 + 1
570   val2 = val2 + 1
571   val3 = val3 + 1
572   val1 = val1 + 1
573   val2 = val2 + 1
574   val3 = val3 + 1
575   val1 = val1 + 1
576   val2 = val2 + 1
577   val3 = val3 + 1
578   val1 = val1 + 1
579   val2 = val2 + 1
580   val3 = val3 + 1
581   val1 = val1 + 1
582   val2 = val2 + 1
583   val3 = val3 + 1
584   val1 = val1 + 1
585   val2 = val2 + 1
586   val3 = val3 + 1
587   val1 = val1 + 1
588   val2 = val2 + 1
589   val3 = val3 + 1
590   val1 = val1 + 1
591   val2 = val2 + 1
592   val3 = val3 + 1
593   val1 = val1 + 1
594   val2 = val2 + 1
595   val3 = val3 + 1
596   val1 = val1 + 1
597   val2 = val2 + 1
598   val3 = val3 + 1
599   val1 = val1 + 1
600   val2 = val2 + 1
601   val3 = val3 + 1
602   val1 = val1 + 1
603   val2 = val2 + 1
604   val3 = val3 + 1
605   val1 = val1 + 1
606   val2 = val2 + 1
607   val3 = val3 + 1
608   val1 = val1 + 1
609   val2 = val2 + 1
610   val3 = val3 + 1
611   val1 = val1 + 1
612   val2 = val2 + 1
613   val3 = val3 + 1
614   val1 = val1 + 1
615   val2 = val2 + 1
616   val3 = val3 + 1
617   val1 = val1 + 1
618   val2 = val2 + 1
619   val3 = val3 + 1
620   val1 = val1 + 1
621   val2 = val2 + 1
622   val3 = val3 + 1
623   val1 = val1 + 1
624   val2 = val2 + 1
625   val3 = val3 + 1
626   val1 = val1 + 1
627   val2 = val2 + 1
628   val3 = val3 + 1
629   val1 = val1 + 1
630   val2 = val2 + 1
631   val3 = val3 + 1
632   val1 = val1 + 1
633   val2 = val2 + 1
634   val3 = val3 + 1
635   val1 = val1 + 1
636   val2 = val2 + 1
637   val3 = val3 + 1
638   val1 = val1 + 1
639   val2 = val2 + 1
640   val3 = val3 + 1
641   val1 = val1 + 1
642   val2 = val2 + 1
643   val3 = val3 + 1
644   val1 = val1 + 1
645   val2 = val2 + 1
646   val3 = val3 + 1
647   val1 = val1 + 1
648   val2 = val2 + 1
649   val3 = val3 + 1
650   val1 = val1 + 1
651   val2 = val2 + 1
652   val3 = val3 + 1
653   val1 = val1 + 1
654   val2 = val2 + 1
655   val3 = val3 + 1
656   val1 = val1 + 1
657   val2 = val2 + 1
658   val3 = val3 + 1
659   val1 = val1 + 1
660   val2 = val2 + 1
661   val3 = val3 + 1
662   val1 = val1 + 1
663   val2 = val2 + 1
664   val3 = val3 + 1
665   val1 = val1 + 1
666   val2 = val2 + 1
667   val3 = val3 + 1
668   val1 = val1 + 1
669   val2 = val2 + 1
670   val3 = val3 + 1
671   val1 = val1 + 1
672   val2 = val2 + 1
673   val3 = val3 + 1
674   val1 = val1 + 1
675   val2 = val2 + 1
676   val3 = val3 + 1
677   val1 = val1 + 1
678   val2 = val2 + 1
679   val3 = val3 + 1
680   val1 = val1 + 1
681   val2 = val2 + 1
682   val3 = val3 + 1
683   val1 = val1 + 1
684   val2 = val2 + 1
685   val3 = val3 + 1
686   val1 = val1 + 1
687   val2 = val2 + 1
688   val3 = val3 + 1
689   val1 = val1 + 1
690   val2 = val2 + 1
691   val3 = val3 + 1
692   val1 = val1 + 1
693   val2 = val2 + 1
694   val3 = val3 + 1
695   val1 = val1 + 1
696   val2 = val2 + 1
697   val3 = val3 + 1
698   val1 = val1 + 1
699   val2 = val2 + 1
700   val3 = val3 + 1
701   val1 = val1 + 1
702   val2 = val2 + 1
703   val3 = val3 + 1
704   val1 = val1 + 1
705   val2 = val2 + 1
706   val3 = val3 + 1
707   val1 = val1 + 1
708   val2 = val2 + 1
709   val3 = val3 + 1
710   val1 = val1 + 1
711   val2 = val2 + 1
712   val3 = val3 + 1
713   val1 = val1 + 1
714   val2 = val2 + 1
715   val3 = val3 + 1
716   val1 = val1 + 1
717   val2 = val2 + 1
718   val3 = val3 + 1
719   val1 = val1 + 1
720   val2 = val2 + 1
721   val3 = val3 + 1
722   val1 = val1 + 1
723   val2 = val2 + 1
724   val3 = val3 + 1
725   val1 = val1 + 1
726   val2 = val2 + 1
727   val3 = val3 + 1
728   val1 = val1 + 1
729   val2 = val2 + 1
730   val3 = val3 + 1
731   val1 = val1 + 1
732   val2 = val2 + 1
733   val3 = val3 + 1
734   val1 = val1 + 1
735   val2 = val2 + 1
736   val3 = val3 + 1
737   val1 = val1 + 1
738   val2 = val2 + 1
739   val3 = val3 + 1
740   val1 = val1 + 1
741   val2 = val2 + 1
742   val3 = val3 + 1
743   val1 = val1 + 1
744   val2 = val2 + 1
745   val3 = val3 + 1
746   val1 = val1 + 1
747   val2 = val2 + 1
748   val3 = val3 + 1
749   val1 = val1 + 1
750   val2 = val2 + 1
751   val3 = val3 + 1
752   val1 = val1 + 1
753   val2 = val2 + 1
754   val3 = val3 + 1
755   val1 = val1 + 1
756   val2 = val2 + 1
757   val3 = val3 + 1
758   val1 = val1 + 1
759   val2 = val2 + 1
760   val3 = val3 + 1
761   val1 = val1 + 1
762   val2 = val2 + 1
763   val3 = val3 + 1
764   val1 = val1 + 1
765   val2 = val2 + 1
766   val3 = val3 + 1
767   val1 = val1 + 1
768   val2 = val2 + 1
769   val3 = val3 + 1
770   val1 = val1 + 1
771   val2 = val2 + 1
772   val3 = val3 + 1
773   val1 = val1 + 1
774   val2 = val2 + 1
775   val3 = val3 + 1
776   val1 = val1 + 1
777   val2 = val2 + 1
778   val3 = val3 + 1
779   val1 = val1 + 1
780   val2 = val2 + 1
781   val3 = val3 + 1
782   val1 = val1 + 1
783   val2 = val2 + 1
784   val3 = val3 + 1
785   val1 = val1 + 1
786   val2 = val2 + 1
787   val3 = val3 + 1
788   val1 = val1 + 1
789   val2 = val2 + 1
790   val3 = val3 + 1
791   val1 = val1 + 1
792   val2 = val2 + 1
793   val3 = val3 + 1
794   val1 = val1 + 1
795   val2 = val2 + 1
796   val3 = val3 + 1
797   val1 = val1 + 1
798   val2 = val2 + 1
799   val3 = val3 + 1
800   val1 = val1 + 1
801   val2 = val2 + 1
8
```


Integers Don't Float

by Ray Mariella

To do graphics programming, you need a fast, integer square root routine—the faster the better. Unfortunately, many published square root routines seem to be transported floating-point routines, and the demands for accuracy are quite different for floating-point numbers and integers. In fact, a single pass of Newton's method may give you sufficient accuracy and still be faster than the routines borrowed from floating-point experience.

The use of Newton's method is familiar: given a target integer, N , the first step is to get an estimate X_0 of the square root of N , then apply Newton's first derivative method to iterate, which is:

$$f(X_1) = f(X_0) + (X_1 - X_0)(df(X)/dX)$$

If \sqrt{N} is desired, let $f(X) = X \times X$ and $df(X)/dX = 2X$. Then, if X_1 is the square root of N , $f(X_1) = N$. Using this definition of $f(X)$, the equation can be written $N = (X_0 \times X_0) + (X_1 - X_0)(2X_0)$, or to find X_1 from a first guess X_0 , use:

$$X_1 = (X_0 + N/X_0)/2$$

If X_1 isn't accurate enough, substitute X_1 for X_0 in this second equation to get X_2 , and so on. The catch comes from the implementation of "and so on."

In algorithms that have been trans-

Ray Mariella, 872 El Quanita Ct., Danville, CA 94526. Ray works at the Lawrence Livermore National Laboratory in Livermore, California, growing compound semiconductor films for ultra-high-speed circuits and optical devices.

Calculating square roots without penalties

ferred from floating-point procedures, in which 64 bits and more are used for accuracy, many iterations may be needed and an error value must be calculated and monitored. When the error is small enough, the routine stops and returns the latest value, X_i , as the square root. Because multiple iterations are needed and an error value is monitored on each pass, the usual code does not use the second equation on its first pass but rather uses $X_1 = N/X_0$ for convenience. You can see this in the typical code fragment for 32-bit integers on the 8086, shown in Example 1, below.

Even with a good first guess of X_0 , such as the average of the maximum

lower bound and the minimum upper bound by factors of 2, a single pass of ISQRT does not usually suffice. For example, to find the square roots of all the integers from 1 to 65,535, an average of 1.6 divisions per root is needed, or an extra 42,150 iterations. Using the same first guess, ROOT.C (the C code in Example 2, page 50) allows the accurate calculation of the integer square roots using a single pass. The assembly-language code for one pass of Newton's method is much shorter than that for ISQRT, as shown in Example 3, page 52.

At this point it's useful to define accuracy. With most integers, an integer square root is not an exact root. When I speak of the "exact" square root, I refer to rounding the exact floating-point square root to the nearest integer. My assembly-language code is written for the 8086 line of CPUs, and my "exact" answers come from the 8087 numeric coprocessor. (See Listings One through Four, beginning on page

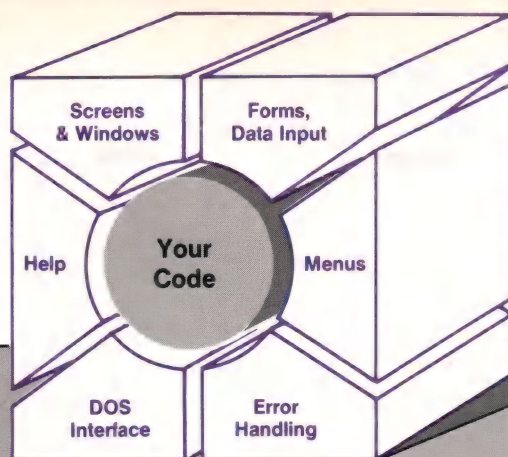
```

; the 32 bit integer N is in DI:SI
; initial guess X0 is in BX
;
ISQRT:    MOV  DX,DI                ;prepare for division
          MOV  AX,SI                ;DX:AX / BX
          DIV  BX                    ;N/X0
          SUB  AX,BX                ;error term
          CMP  AX,1                 ;check if > +1
          JG   ISQ1                 ;if above 1, keep on
          CMP  AX,\sc0\1            ;check for \sc0\1,0,+1
          JGE  done                 ;if OK, get out
ISQ1:     SAR  AX,1                 ;(N/X0 \sc0\X0)/2
          ADD  BX,AX                ;(N/X0 +X0)/2 = X1
          JMP  short ISQRT          ;use X1 as X0

```

Example 1: A code fragment for calculating square roots of 32-bit integers on i6

30 day moneyback
satisfaction guarantee



C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.

C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

Introducing...

C-Worthy® Interface Library

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0+ and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

Tech Specs

- ▶ Compilers: Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- ▶ 350+ functions written in C, 75+ in Assembler.
- ▶ Menus: Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- ▶ Errors: DOS, program, and user.
- ▶ DOS Interface: 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- ▶ Help: System and context sensitive.
- ▶ Screens: Screen display, color palettes, save, restore, scroll, more.
- ▶ Windows: Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- ▶ Keyboard Handling: Regular, function, interrupt, background procedures.
- ▶ Editing: String and word wrap text.
- ▶ Form Interface Library: 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- ▶ Foreign Languages: All text messages in separate files for easy translation.
- ▶ Compatible with MS Windows.
- ▶ OS/2 special overlay when released.
- ▶ Machines: Autodetect for MDA, CGA, EGA, VGA, TI, AT&T, Victor.
- ▶ No royalties.

"I heartily recommend this package."

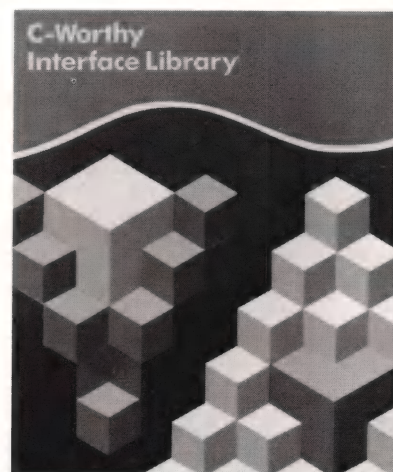
— David A. Schmitt, president, Lattice, Inc.
Over 400 developers in 16 countries already use it.

Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

CIRCLE 127 ON READER SERVICE CARD

"C-Worthy is a comprehensive C library whose time has come. I heartily recommend it as your next purchase." —Computer Language, 8/87



C-Worthy Interface Library:

Object only	\$ 195
Form Interface Library add-on	\$ 100
Object with Forms	\$ 295
Object with Forms & Library Source	\$ 495

Please specify compiler and version when ordering.

To Order Call

(800) 821- 2492

in MA (617) 337-6963

**Solution
Systems™**

541-D Main Street, Suite 410
South Weymouth, MA 02190

INTEGERS DON'T FLOAT (continued from page 48)

98.)

Applications vary, and if you

demand that the integer square root either agree with the exact root or differ by -1, then ROOT.C works up to N=127,087 (the square root of 127,088 is 356.49 and a single pass

gives 357). If you're willing to accept 0, or +1, or -1 as an error, then a single pass of Newton's method can be used up to 1,941,799 (for 1,941,800 the square root is 1393.48 and a single pass gives 1,395).

Skipping all those compare, conditional jump statements, and multiple passes can save a lot of CPU time. On my 8-MHz PC6300 with V30, the assembly-language program RALL16 (Listing Four) which is optimized for 16-bit integers, finds all the square roots from 1 to 65,535 in 2.5 seconds, with an empty loop time of 0.2 seconds, or about 35 microseconds per root. This is slightly faster than the speed at which the 8087 performs the same task. I found that 29,776 of the roots were 1 less than the exact root and the rest agreed with the exact root. The Microsoft C 4.0 version of RALL16 took 4.6 seconds for all integers from 1 to 60,000, or about 73 microseconds per root.

For 32-bit integers up to 4,294,967,295 (FFFF:FFFF), two passes of the second equation are needed. My program code, ISQRT32.ASM (in

```

/* ROOT.C a square root algorithm by RPM */
/* long integers, single pass of Newton */

#include <stdio.h>

main (\scl28\ )
{
    long int N, guess2, sqrrt;
    register int infi, guess1;

    printf ( "\n square root of what number " );
    scanf ("%ld",&N);
    { guess1 = infi =1;
      guess2 = N;
    logit: infi <= 1;
      if ( infi < guess2 )
      { guess2 >= 1; /* div by 2 */
        guess1 = infi;
        goto logit;
      }
      guess1 += guess2; /* sum */
      guess1 >= 1; /* avg */
    /* newton's method */
      infi = N / guess1;
      sqrrt = infi + guess1;
      sqrrt >= 1;
    }
    printf ( " square root = %ld", sqrrt);
}

```

Example 2: C code to calculate integer square roots using a single pass



Now you can use QPARSER+ to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc.

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.

Another translation by QPARSER+

Just \$475 (PC/XT/AT) — **FREE** demo disk available

QCAD Systems

1164 Hyde Avenue, San Jose CA 95129 (408) 727-6884
Outside Calif. call TOLL-FREE (800) 538-9787

CIRCLE 128 ON READER SERVICE CARD

M Street Software

80386 Support!

SCRUTINY

Advanced symbolic debugger.

- Multi-language: compatible with Turbo Pascal, Microsoft Assembler, others.
- Multi-DOS: works with all MS-DOS/PC-DOS computers.
- Multi-level: debug at source level and machine level, separately or together.
- Multi-display: debug character-mode and graphics-mode programs, with movable debug windows.
- Multi-chip: support for 8086, 80186, 80286, 80386.
- Fast 80386 "memory breakpoints" (stop program when specified variable is accessed or modified).

Scrutiny/Master \$99.95

for debugging Turbo Pascal, Microsoft Assembler, and other languages.

Scrutiny/Turbo Special price! \$49.95

for debugging Turbo Pascal only.

VISA/MC AMEX accepted. In Texas please add sales tax. Outside of North America add \$10 per item shipping.

M Street Software

5400 E. Mockingbird Lane Suite 114

Dallas, Texas 75206

214-827-4908

Information also available via our 24 hour 300/1200 modem: 214-669-1882.

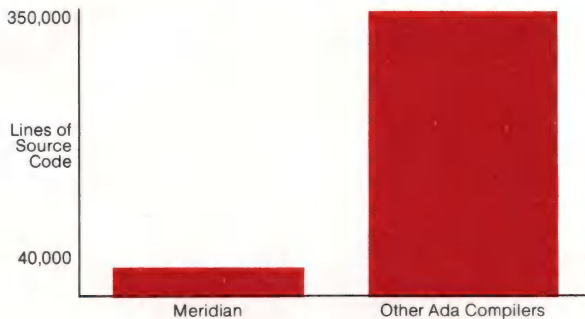
CIRCLE 129 ON READER SERVICE CARD

THE ADA[®] WORLD HAS CHANGED.[™]

Next Generation Ada Technology

Third generation Ada is here today with Meridian AdaVantage, and it's validated by the Department of Defense and their suite of 2,700 tests. Ada's first generation was the proving ground technology of the early eighties. Then came the large validated compilers of the mid-eighties, which were often inefficient and machine specific. Now Meridian offers a compact efficient Ada technology that is highly portable from PC's to minis to mainframes.

COMPACT MERIDIAN Ada IMPLEMENTATION



The chart above depicts the dramatic difference between Meridian's implementation and other commercially available Ada implementations. Meridian's order of magnitude smaller code size results in an extremely fast compiler that will produce highly optimized code.

Meridian's Track Record

Since 1981, we've been building compilers the old-fashioned way, through hard work and experience gained from our development of successful portable compiler products. These products are installed at over 1,500 sites, including almost all major DoD contractors as well as commercial software developers.



THIS PRODUCT CONFORMS
TO ANSI/MIL-STD-1815A AS
DETERMINED BY THE AJPO
UNDER ITS CURRENT
TESTING PROCEDURES

Ada is a registered trademark of the U.S. Government (AJPO). AdaVantage and AdaStarter are trademarks of Meridian Software Systems, Inc. References to other computer systems use trademarks owned by the respective manufacturers.

Price/Performance Breakthrough

The Meridian AdaVantage v2.0 validated Ada compiler costs \$795 and provides a production quality Ada development tool.

SIEVE BENCHMARK

	Meridian v2.0	Alsys v1.3
Compile and link time	27 sec	59 sec
Execution time	4.6 sec	4.9 sec
Execution size	27,344 bytes	42,129 bytes
Price	\$795	\$2,995

NOTE: All times measured on an IBM at 5170 (8MHz) and a 4MB RAM card required by the Alsys system. When running without the RAM card, the Meridian compile and link time is 46 seconds.

The Meridian AdaStarter incorporates all of the features of AdaVantage, with certain size limitations. AdaStarter is perfect for anyone who wants to learn how to program in Ada. The complete \$99 cost is applicable toward the purchase price of the AdaVantage production compiler.

The compilers all run in a standard PC configuration with 640K of memory, a hard disk, and DOS v2.1 or higher.

To order today, or get more information, call 1-800-221-2522 (outside California) and 1-714-380-9800 (inside California).

"The Meridian compiler is a very well-thought-out compiler. The compiler is fast and execution speed is more than adequate... Overall, we'd rate the Meridian compiler as very solid."

— COMPUTER LANGUAGE, DECEMBER 1986

"The more affordable AdaVantage v1.0 is good for the average programmer because of its price, the extent of its implementation, and its relaxed hardware requirements... [AdaVantage v2.0] should be competitive with Alsys Ada, since it will be a full Ada at less than a third of the price of Alsys."

BYTE, JULY 1987



23141 Verdugo Drive, Suite 105, Laguna Hills, CA 92653
800/221-2522 (outside CA) 714/380-9800 (inside CA)
Telex: 650-268-0547 MCI Fax: 714/380-1683

INTEGERS DON'T FLOAT (continued from page 50)

Listing One), for the 8086 line is cumbersome, but it does work for all these integers (it takes approximately 153 hours to compare the V30 and 8087 square roots). Again, the result after two passes of Newton's method either agrees with the exact integer or is 1 less. This code executes in about 110 microseconds per root on my machine. (Here the 8087 is more than twice as fast as the V30 and looks very attractive indeed!)

Again with Microsoft C 4.0, the two-pass version of ROOT.C takes an average of 360 microseconds per

root for integers from 10,000 to 100,000,000 in steps of 10,000 (using register variables). This V30 performance is about the same speed as the same program on a Macintosh Plus with Lightspeed C. The maximum integer that ROOT.C can handle depends upon the compiler and its acceptance of such things as unsigned register variables.

The decision whether to use the single-pass or double-pass version of Newton's procedure can usually be made by the programmer before starting, based on the number of pixels on the screen and thus the maximum integer that the program will see. In either event, my measurements have shown that using the

unaltered Newton's method provides greater speed and somewhat better accuracy than can be attained using the more common algorithms that came from the floating-point literature.

Along the same lines, in the March 1986 issue of DDJ, Richard Campbell described both a C routine and an assembly-language version for the NS320xx processor for calculating square roots.—eds.

Availability

Most of the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 98.)

Vote for your favorite feature/article.
Circle Reader Service No. 4.

```
; the 32 bit integer N is in DI:SI
; initial guess X0 is in BX
;
NEWTON:  MOV  DX,DI          ;prepare for division
         MOV  AX,SI          ;DX:AX / BX
         DIV  BX             ;N/X0
         ADD  BX,AX          ; (N/X0 +X0)/2 = X1
         RCR  AX,1           ; (N/X0 \sc0\X0)/2
```

Example 3: Assembly-language code for one pass of Newton's method

4 Times Faster than MASM 5.0

All MASM features (except 386 & CodeView support)

Plus

- *Generates smaller code! (no inserted NOP's)
- *Automatically handles jumps out of range
- *Handles most of MASM's phase errors
- *Built in MAKE
- *Up to 15,000 symbols
- *Simplified segmentation

OPTASM
\$195



1622 N. Main St., Butler, PA 16001
(412) 282-0864 (800) 833-3061
TELEX 559215

CIRCLE 131 ON READER SERVICE CARD



Our front end helps protect your back end.

Today's users require sophisticated interfaces for their applications. Yet complex front ends are a real pain to create, especially when the specs change and your deadlines don't.

But now JYACC introduces JAM™, a powerful user interface development tool that makes it easier than ever to design and revise your complex applications.



Use windows and colors freely with JAM.

JAM is the first tool that does it all, from prototyping to implementation. With JAM, you start by creating screens and linking them together to develop an application shell. You can experiment with the look of the interface, and even explore "what if" scenarios on the application flow. Then you attach processing routines, and your application is complete.

JAM works under the following operating systems:

- UNIX®
- MS-DOS®
- VMS®
- XENIX®
- RMX™
- VOS™

You'll be amazed at how quickly your applications spring to life with JAM's interactive screen management utility. You can use features like context-sensitive help, shifting and scrolling fields, a variety of visual attributes and extensive data validations to create exciting screens, windows and menus—all without writing a single line of code. JAM also lets you test your prototypes at any time.

JAM lets you draw from its extensive subroutine library to help you write processing routines faster. And revisions to your applications are easier,

because your subroutines are insulated from the data entry and presentation details of the interface.

JAM is extremely portable. It runs on almost every computer, from PCs to superminis, and works under 6 operating systems. This allows you to develop consistent interfaces throughout your company—a significant asset to the Fortune 500 companies that have been using JAM for more than a year.



Enhance applications with context-sensitive help.

JAM from JYACC. It gets your front ends—and back ends—in great shape. Call for more information about JAM and our demo diskette. **800-458-3313** JYACC FORMAKER™, JAM's screen manager, is also available separately. JYACC, Inc., 116 John Street New York, NY 10038 212-267-7722

Introducing JAM

JYACC Application Manager. *The Composer for Sophisticated Applications.*

JYACC

Excellence in Systems & Application Design

MS-DOS, XENIX: Microsoft Corp.; UNIX: AT&T Bell Labs; RMX: Intel Corp.; VMS: Digital Equipment Corp.; VOS: Stratus Corp.

CIRCLE 132 ON READER SERVICE CARD

A Graphics Toolbox for Turbo C—Part 2

by Kent Porter

In Part 1 of "A Graphics Toolbox for Turbo C" (DDJ, November 1987), I developed a library of low-level graphics routines for Turbo C and explored ways to incorporate them into high-level drawing routines for APA (all-points addressable, or pixel-oriented) graphics. In this article, I use the library routines developed in Part 1 to create text graphics such as menu bars, pop-up windows, pull-down submenus, and the like.

Popping up a visual object that obscures a portion of the screen, then restoring that overlaid area after removing that object, is a visually exciting if relatively simple programming trick. Before writing an object to screen, you simply copy display memory into a save buffer; to make the object go away—that is, to restore the screen to its former appearance—you just write back the screen previously saved to buffer into display memory. In this way, the text obscured by the pop-up reappears magically intact.

Finding display memory can be a bit more challenging, however. The display memory location depends on the kind of adaptor you're using. To get that information from the operating system, I constructed a function called *videomode()*. It determines which type of video adaptor is active: if *videomode()* returns 7, there's an MDA with display memory at *B000h:0*; otherwise, dis-

Easy menu bars, pop-up windows, and pull-down submenus

play memory begins at segment *B800h*.

Text screens for IBM-standard adaptors occupy 4,096 bytes, whereas for the Hercules the display memory is 16,384 bytes. You can find out the video buffer size from *40h:4Ch* using Turbo C's *peek()* function, which returns an integer indicating screen memory size.

Some non-MDA adaptors provide for more than one video page. That is, they subdivide the total display memory into 4K segments (pages) and allow you to select which is the active display buffer. That's what the video library function *setpage()* does, and *activepage()* tells which it is. Therefore, with a non-MDA adaptor, you have to determine the active page and multiply by 4,096 to determine the starting offset of the current video buffer within segment *B800h*. (Or you can fetch this number directly from *40h:4Eh*.)

Once you know the size and location of the active page's video buffer, you can allocate a node of that size on the heap and use Turbo C's *move-data()* function to save the display image. You can then create a pop-up. Later, simply reverse the source and destination parameters in *move-data()* to restore the screen and make the pop-up vanish.

Restoring the screen is problematic with the IBM-brand CGA,

which has a flawed design that generates "snow." This occurs when you write directly to screen memory in text mode. Fortunately, this isn't a problem with the Compaq and most non-IBM CGA clones. The solution with the IBM board involves synchronizing character movements with the video controller. (For more detail on this procedure, see Ray Duncan's *Advanced MS-DOS*, page 79).

The *saveScrn()* and *restScrn()* functions near the end of *POPWIN.I* (Listing One, page 106) furnish somewhat simplified versions of this discussion. They assume that the active page is 0—the default—and that an IBM-standard adaptor is in use. Also, they don't deal with snow.

You might wish to take a more sophisticated approach to saving screens. In the demo developed for this article, I never save more than one screen image, and thus I simply copy the active screen buffer to and from an object of the same size on the heap.

Real-world situations often have a hierarchy of pop-ups, in which one leads to another, to another, and so on. In that case, you build a stack (LIFO) structure on the heap so that you can retreat back down the hierarchy in an orderly manner. A circular doubly linked list is ideal for this purpose.

In summary, then, the steps for handling pop-ups are:

- Save the display buffer on the heap.
- Write the pop-up.
- On a signal from the user, copy the heap image back to the display buffer, thus restoring the screen.

Kent Porter, 1909-4 Montecito Rd., Mountain View, CA 94043. Kent has written 17 books about programming and hundreds of magazine articles on computer hardware and software. He is a technical editor for DDJ.

Now that we've solved one of the big mysteries surrounding pop-ups, let's discuss the abstraction of complexity.

Visual Objects as Data Structures

Although the screen is a free-form area that can contain literally anything you want it to, special objects such as pop-ups and menu bars always conform to a prescribed set of attributes. That is, they have a defined number, size, location, color combination (foreground/background), border, text content, and so on.

The point is that any object can be defined in terms of two characteristics: its appearance and its text content. Sometimes it's appropriate to combine them into one, and sometimes it's not. I'll consider both cases.

This approach suggests the use of data structures and accompanying routines that interpret those structures on the display. On that basis, you can describe visual objects by building structures and leave it to the routines to translate them into reality. Let's prove the point with a discussion of menu bars.

Building Menu Bars

The menu bar has become a staple of interactive software. You see it everywhere: Reflex, Paradox, Lotus 1-2-3, Turbo C itself. It's that row of selections across the top of the screen, telling you the categories of things you can do with the program. Usually, when you pick a selection, that choice leads to others via another menu bar (the Lotus approach) or a pull-down submenu, which I discuss later in this article.

In software with a consistent user interface, the menu bar always has a predefined place on the screen. Here I assume it's at the top—row 0—but your own application might dictate another location, in which case you'll have to modify the *menubar()* routine to suit. A menu bar also has a fixed text content. If you want to add or delete selections, that's a different menu.

The advantage of a routine such as *menubar()* (Listing Two, page 106) is that it interprets the structure passed to it. This enables you to

pass different structures to a common function, obtaining the desired results in each case. The previous menu bar is replaced by the new one, effecting an instantaneous change in context.

Given that all menu bars in an application occupy the same display row, any given menu bar can be described in terms of its color scheme, number of selections, and text content. That leads to the definition of a menu bar specification

**With multiple
pop-up windows,
a circular
doubly linked list
is ideal for
screen saving.**

as the C structure:

```
typedef struct {
    int background, foreground; /*
                                   colors */
    int nsels; /* no. of selections */
    char *sel; /* pointer to text content */
} MENUBARSPEC;
```

You might want to use different color schemes to identify various menu bar levels: for example, top level = red background, middle level = magenta, and low level = blue. Also, it's likely that each menu will have its own number of potential selections. A generalized routine such as *menubar()* can easily handle these variations, simply plucking them from the structure.

The *menubar()* function first identifies its environment by getting the active page and the number of text columns available (40 or 80), and then it builds the background/foreground attribute byte with a call to the video library's *chattr()* function (a service function that does not call the ROM BIOS directly or indirectly). Next, it clears the current menu bar row and sets the attribute by writing

out a stream of spaces for the full screen width. Finally, it writes the text of the menu selections.

The menu selections are specified in a separate string, a pointer to which appears in the menu specification. For convenience, the string has the form:

```
menutext = "sel1\0sel2\0sel3\0seln\0"
```

in which each text element is null-terminated. This is easier to handle than the more conventional two-dimensional array of characters, as the last loop in *menubar()* illustrates.

Using this approach, you can define a menu bar by first initializing its text content, then its appearance and a pointer to the text content in a structure of type *MENUBARSPEC*. If you had two menu levels, separately defined, you could display the top level with *menubar(&firstLevel)*; and, on the appropriate signal from the user, switch to the next with *menubar(&secondLevel)*.

Because menu bars tend to have a fixed appearance and content, it's easy to implement them using this scheme. Pop-ups are more flexible and consequently a little more demanding. Still, you can create a wide variety of pull-down menus and pop-up windows using a common set of routines and structures, as discussed next.

Pop-Ups and Pull-Downs

In practice, there's no difference between the two. A pop-up is a window that can appear anywhere on the display. A pull-down is a pop-up that looks as though it dropped down from a menu bar selection. The distinction is purely one of location.

Pop-ups rely on the ROM BIOS window routine (function 6 under interrupt 10h). Despite its comprehensive-sounding name, the BIOS routine is useful for only two things: scrolling a subset of the screen and filling that subset with a background attribute to give it a color. User-written routines that want to do more than this will have to provide code to overcome its limitations.

POPWIN.I (Listing One) furnishes generalized routines for managing

pop-up windows, relying on the low-level routines from the video library. This is an `#include` file, and there's little advantage to making it into a linkable library because any windowing program needs all the functions it contains.

Like a menu bar, a pop-up can be described in terms of its characteristics, which in this case are:

- the coordinates of its opposite corners (giving both location and size)
- the text attribute (background/foreground colors)
- the border type (single- or double-score)

The `POPDESCR` structure in `POPWIN.I` provides a window descriptor.

The static array `bord[]` gives the ASCII values for single- and double-score text characters that form the window border. The border choices are 0 for no border and 1 and 2 for single- and double-scores, respectively.

`POPWIN.I` contains four functions for directly manipulating windows:

- `popMake()` creates and displays the blank window described by the structure passed to it as an argument
- `popScroll()` scrolls the window's contents upward by one row
- `popxy()` positions the cursor relative to the window's upper-left corner
- `popPuts()` writes a string starting at the specified cursor position

The other two functions—`saveScrn()` and `restScrn()`—save and restore the display image.

To use these routines, you must first initialize a `POPDESCR` structure for each window you plan to use. Thereafter, simply pass a structure address to the pop routines as you call them. Because the routines do not preserve the caller's cursor position, your code must do so before the calls.

Note that, unlike the menu bar structure, `POPDESCR` doesn't contain a pointer to the window's text contents. This allows you to use a

common descriptor for several windows that contain variable text: context-sensitive help windows, for example. Use `popPuts()` to write to the window.

The `popMake()` function opens a window, fills it with the assigned text attribute, draws a border around it (unless the structure's border member is zero), and positions the cursor in the upper-left corner. When creating a new window, call this function first. Note that the border is outside the window's defined text area; that is, if the window's upper-left corner is at (12, 15), the left side of the box is at column 11 and the top is at row 14. The right side and bottom are similarly one unit beyond the text area. Thus, don't put a window at any extreme position on the display, lest the border be partly invisible or partially visible in an unexpected place.

The `popScroll()` routine shifts the window's contents upward one row by calling the `winScroll()` function in the video library, passing it the boundaries of the area to be scrolled and the text attribute to fill the newly opened row at the bottom. `PopPuts()` uses this routine to guard against text overrunning the bottom of the window.

Use `popxy()` to position the cursor within the window, using the upper-left corner as the origin. No matter where the window is physically located, `popxy(0,0,win)` refers to its upper-left corner, effecting viewport text coordinates. This function is therefore a window-oriented analog to the video library's `gotoxy()`.

The `popPuts()` function writes text to the window, starting at the specified coordinates with respect to the window's upper-left corner. It contains safeguards to ensure that the text does not get outside the window, wrapping if the text tries to go beyond the right side and scrolling if it tries to go past the bottom. The function accommodates a new line (`'\\n'`) embedded in the text, but it does not have formatting capabilities like `printf()` does. If you want to write variables to a window, use `sprintf()` to set up the text string, then pass the results to `popPuts()`.

The functions for saving and restoring the video buffer work unpre-

The C Programmer's Assistant

C TOOLSET

Unix-like Utilities for Managing C Source Code

No C programmer should be without an assistant. C ToolSet provides you with the support you need to make C programming easier.

All of the utilities are tailored to the C language but you can modify them to work with other languages as well.

Source code in standard K&R C is included. You are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

The documentation contains descriptions of each program, a listing of program options, and a sample run. On-line help responds to `?` on the command line with a list of options

12 Time Savers

DIFF - Compare files line by line; use CMP to compare byte by byte.
GREP - Regular expression search.
FCHART - Trace the flow of control between large program modules.
PP - Format C program files so they are easier to read.
CUTIL - General purpose file filter.
CCREF - Cross reference variables.
CBC (curly brace checker) - Check for pairing of curly braces, parens, quotes, and comments.
Other utilities include **DOCMAKE**, **ASCII**, **NOCOM**, and **PRINT**.

Requires MSDOS and 12K RAM

MONEYBACK GUARANTEE

Try C ToolSet (\$95) for 30 days — if not satisfied get a full refund.

Call (800) 255-4659

In MA (617) 331-0800



The Coder's Source™

541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE 133 ON READER SERVICE CARD

TURBO PASCAL AND TURBO C... MEET

TURBOHALO

"Ideal! TurboHALO does the job
comparable to packages costing
\$3000 to \$4000."

Jim Bromley
Superintendent of
Spectrum Management

"I like the speed of
TurboHALO...it's ten times faster
than the competition."

Deniz Terry
Doctoral Candidate

"TurboHALO is so fun...
I use it to design
programs as a hobby...It's got
lots of ability."

William Porter
Control Systems Manager

"We evaluated all of the graphic
development packages for Turbo
Pascal, and TurboHALO was the
hands down winner!"

Quinn Curtis
Largest New England Distributor

It's time to put graphics into your programming.

A picture's worth a thousand words.
So your programming isn't complete
until you have graphics. TurboHALO
brings your screen and printer to life
with subroutines that draw, chart, map,
and display. All with color, shape,
clarity, perspective and motion. With
TurboHALO, create any picture you
can imagine.

TurboHALO gives you graphics power.

TurboHALO gives you everything you
need for Turbo C and Turbo Pascal
graphics programming. A library of
over 150 graphics subroutines. Drivers
for over 42 graphics hardware devices
for the IBM PC family and compatibles.

You can create the images you want,
on the hardware you have!

Fast, proven and reliable.

TurboHALO is up to ten times faster
than other graphics toolkits. And with
TurboHALO you get proven, reliable
programming tools used by
professionals for years.

You'll like TurboHALO or your money back!

TurboHALO is available for only \$95.

You get an unconditional 45-day
money-back guarantee on TurboHALO.

To order, call your dealer or IMSI at
(415) 454-7101 or (800) 222-4723; in
CA (800) 562-4723; in Washington DC
(202) 363-9340 or in NC (919) 854-4674.

FOR GRAPHICS PROGRAMMING

TurboHALO requires 256K memory (min); memory resident drivers require 2K (Turbo Pascal only); DOS version 2.0 or higher; Borland language compiler required. TurboHALO is a trademark of Media Cybernetics and IMSI. Turbo C and Turbo Pascal are trademarks of Borland.

YES, I want to see the difference
TurboHALO makes in my graphics
programming! Rush me the following
TurboHALO Graphics Toolkit(s) @ \$95
each plus \$3 shipping. California
residents add 6% sales tax.

- ☐ TurboHALO for Turbo Pascal
☐ TurboHALO for Turbo C
☐ Check enclosed for \$ _____
(made payable to IMSI)
☐ Charge my credit card
for \$ _____ ☐ VISA ☐ MasterCard

Signature _____
Card Number _____ Expiration Date _____
Name _____
Title _____
Company _____
Street _____
City _____ State _____ Zip _____

Imsi™

Mail to:
IMSI, 1299 Fourth Street, San Rafael, CA 94901

DDJ12/87

PROGRAMMER BUYS INBOARD 386; ENDS UP IN HOT WATER.

When Jim Johnson's C program got to be more than his IBM PC AT[®] could handle, he had two choices:

Come up with \$6000 for a brand new 386 system.

Or spend ~~\$2000~~ ^{\$1595} for an Intel Inboard[™] 386 system.

As you might suspect, he chose the Intel Inboard 386. And got the full power of a 386 system.

Allowing him to cut his compiling time in half. Without compromising one bit on reliability. And with the extra ~~\$4000~~ ^{\$4700}, he put a mahogany hot tub on his deck.



How can we give you the same for so much less? Simple. If you have an IBM PC AT or compatible, you already own 2/3 of a 386 system. And when you install an Inboard 386, you get the rest of it.

Besides the price, how do they compare? According to a recent *InfoWorld* product review, "Inboard has the best computing speed of all micro systems we have tested to date," including all other 386 systems.

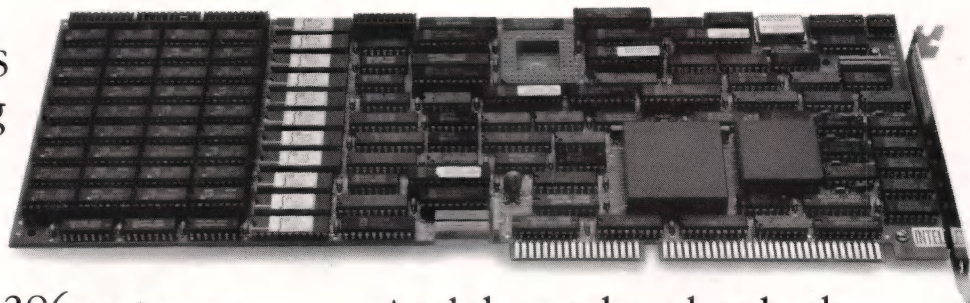
The reason Inboard is so fast is because of its zero wait state cache and 32 bit memory. To give you even greater performance, there's also a special socket for the 80387-16 math coprocessor.

Software compatibility is

unsurpassed as well. According to *PC Week*, "The Inboard 386 proved perfectly compatible with a standard IBM PC AT and every software product we tested." It's compatible with advanced software, too, including 386 control software for multitasking. And a number of developers are already using Inboard to create OS/2® applications.

Of course, you're probably wondering if a ~~\$2000~~ ^{\$1595} system can be as reliable as a \$6000 one. Absolutely. Because it's built by the same company that designed the 80386 microprocessor. And backed by a five-year warranty.

If you still need more information on which system is best, call us at (800) 538-3373. Or contact your local dealer.



And then take a hot bath. When your workload is too much to handle, it's the best way we know to unwind.

intel®

Trademarks/owner: IBM PC AT, OS/2/International Business Machines Corporation; Inboard/Intel Corporation. © 1987 Intel Corporation

CIRCLE 135 ON READER SERVICE CARD

SAVE YOUR PC FROM EARLY RETIREMENT.

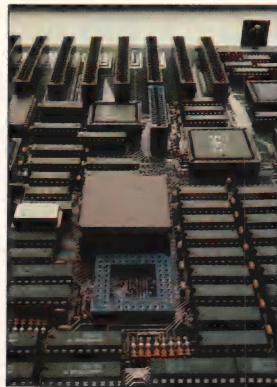


GET HAUPPAUGE'S NEW 386 MOTHERBOARD.

386 SPEED—ONLY \$1,495

Give your PC a new lease on life! With our industry first 386 MotherBoard, your PC, PC/XT or compatible will revel in speeds equal to the Compaq DeskPRO 386. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will knock you off your rocker.

To keep retirement at bay, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — allowing you to run the new generation of DOS, OS/2. We've also included a 16-bit expansion slot that accommodates the latest I/O expansion card. No accelerator card can give you so much versatility.



With 386 power and true AT software compatibility, your business, desktop publishing and engineering applications will get a boost to boast about! **Technical Features** ■ 16 MHz 80386 ■ 1 Megabyte of 100 nsec 4-way interleaved RAM ■ PC/AT compatible I/O and BIOS for support of OS/2 ■ Seven 8-bit expansion slots ■ Two 16-bit expansion slots ■ One 32-bit RAM expansion slot ■ Optional 16 MHz 80387 math coprocessor (\$695)

Put the power of the 386 into your IBM PC for 1/4 the cost of a 386 computer. And put off your PC's retirement. For more information on our easy-to-install Motherboard, call **1 (800) 443-6284**. In New York, call **(516) 360-3827**.

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway,
Commack, New York 11725

Hauppauge!

Trademarks: IBM PC, XT and AT are trademarks of International Business Machines Corporation. Compaq DeskPRO 386 is a trademark of Compaq Computers.

CIRCLE 136 ON READER SERVICE CARD

dictably with IBM display adaptors (MDA, CGA, and EGA) but work normally with Hercules and other third-party video boards. You can make them more intelligent with the techniques in last month's installment. As written, these functions assume that if the video mode is 7, an MDA (or an EGA emulating an MDA) is present and using screen memory at segment *B000* and that otherwise a color device is using the video buffer at segment *B800*.

In either case, because these routines service text displays, they assume a 4,096-byte display memory. *SaveArea()* allocates a node of that size on the heap and copies the video buffer to it, returning a pointer to the node. Afterward, you can create a pop-up. To make the pop-up disappear later, pass the node pointer to *restScrnr()*, which moves the saved image back into the video buffer and frees the heap space.

Now let's put these two visual subsystems—menu bars and pop-ups—to work in a demonstration.

Tying It Together

MENUEMO.C (Listing Three, page 106) writes a menu bar across the top of the screen and provides two pull-down submenus on successive keypresses. Some text appears under the pull-downs so that you can see how overlaid information is restored when a window is removed from the screen.

The pull-downs are described by the *filePop* and *editPop* structures and their text contents by the strings *fileMenu* and *editMenu*. For convenience, the strings contain embedded new lines that will be processed by *popPuts()* to give the pull-downs their proper appearance.

The initialization steps in *main()* use the video library's *chattr()* function to complete the *filePop* structure, then copy the structure to *editPop*. The location of *editPop* is computed with reference to the second entry on the menu bar, and the size of the box is changed. The program is now ready to run.

Setmode(3) has no effect on an MDA, but it places other adaptors

in color mode (an EGA with a monochrome monitor displays the colors as intensities). After clearing the screen, the program writes the menu bar and prints out some information about the display.

The program uses *oldx* and *oldy* to retain the cursor position before reporting it. This is so that, later, a subsequent output line will overlay the cursor position report.

When the user presses a key, the first pull-down appears, created by *popFileMenu()*. On the second keypress, the file pull-down disappears, the background is restored, and *popEditMenu* flashes up the edit pull-down. Two additional keypresses restore the original screen and end the program.

Note that, except for the structure initialization, it takes only one statement to display a menu bar and only two to create a pop-up (*popMake()* and *popPuts()*). These calls exercise a good deal of underlying code, of course, but this code remains out of sight in the *#include* files, where it doesn't needlessly clutter the program's listing.

Full Circle

Obviously, the graphics library I've built here wasn't intended to be a full-blown graphics toolkit; rather, it was meant to provide you with a skeletal framework to build upon. Use the techniques illustrated here to build more powerful visual subsystems in Turbo C that satisfy your specific application development needs.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 106.)

Vote for your favorite feature/article.
Circle Reader Service No. 5.

Introducing
Network Version!

SEIDL VERSION MANAGER

Now SVM supports local area networks and tracks source revisions made by multiple users in both single-site and multi-site configurations.

Plus...

- Archive Database Tracks Source (and Binary) File Revisions
- Audit Trail Reporting Provides Info on Project's Development
- Revision Branches Allow Multiple Courses of Development
- Revision Merging and Deleting Provide Flexibility in Archive Maintenance
- User IDs, Privilege Settings & Passwords Help Resolve Access Conflicts and Maintain Project Integrity
- Optional Text Compression Reduces Storage Requirements
- Menu Driven Shell Makes SVM Easy to Use
- Single-Site: \$299.95*
- 5-site LAN: \$1000 (extendible)

Now Builds
Dependencies!

SEIDL MAKE UTILITY

New program, called *SMKgen*, automatically constructs a dependency file by analyzing the files in a project.

Plus...

- Structured Language Used to Define Dependencies
- Rich Command Set with Over 20 Different Statements
- Ability to Handle Nested Include Files and Library Dependencies
- Performance & Functionality not Found in UNIX Make or Clones
- SMK Only: \$99.95*
- SMKgen: Add \$50.00

CALL TODAY

1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited

*Plus postage and Handling

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 137 ON READER SERVICE CARD

RAM-CACHE MANAGER

Listing One (Listing continued, text begins on page 30.)

Listing 1

```
/* cache header */

typedef struct CACHE {
    int    recl;          /* record length */
    int    maxr;          /* number of records in cache */
    long   hits;          /* number of true finds for cacfind() */
    long   miss;          /* number of misses for cacfind() */
    long   adds;          /* number of calls to cacnew() */
    int    (*proc)();      /* pointer to function for processing */
    long   idnt;          /* identifier passed to proc() */
    long   *nums;         /* numbers of records */
    char   *recs;         /* pointer to first record */
    short  *next;         /* array of next pointers */
    short  *prio;         /* array of prior pointers */
    char   *mark;         /* array of process-record markers */
    short  lru;           /* oldest index */
    short  mru;           /* newest index */
} CACDS;

long    cacamem; /* amount of memory allocated */

CACDS   *cacallo(int, int, int (* )(), long);
char     *cacmem(int);
char     *cacnum(struct CACHE *, long);
char     *cacold(struct CACHE *);
int       cacflsh(struct CACHE *);
int       cacnew(struct CACHE *, short);
char     *cacfind(struct CACHE *, long);
int       cacproc(struct CACHE *, long);
int       cacunpc(struct CACHE *, long);
int       cacstat(struct CACHE *, long *, long *, long *);
int       cacfree(struct CACHE *);
```

End Listing One

Listing Two

Listing 2

```
/* cache.c - memory cache

alan deikman 12/86

These routines handle a memory cache of fixed sized records.
All memory is allocated through the standard library routine
malloc().

Each routine other than cacallo() takes as the first parameter
a character pointer that is originally returned by the cacallo()
routine.

cacallo()      Allocate cache
cacold()       Get oldest record
cacnum()       Number oldest record and make it the newest
cacflsh()      Process all marked records
cacfind()      Find record n and make it newest
cacproc()      Mark record n and make it newest
cacunpc()      Unmark record for processing when freed
cacstat()      Get cache statistics
cacfree()      Free cache

*/

#include <stdio.h>
#include <malloc.h>
#include "cache.h"

/* cacallo() - Allocate cache */

CACDS *cacallo(num, recl, extf, idnt)
int    num;          /* number of records to allocate */
int    recl;         /* length of each record */
int    (*extf)();    /* pointer to external processing function */
long   idnt;         /* parameter passed to external function */
{
    CACDS *cac;
    char   *cacmem();
    int    i = 0;

    /* set up cac structure with initial values */

    if (num < 2) num = 2;
    cac = (CACDS *) cacmem(sizeof(CACDS));
    cac->recl = recl;
    cac->maxr = num;
    cac->proc = extf;
    cac->idnt = idnt;
    cac->hits = 0;
    cac->miss = 0;
    cac->adds = 0;
    cac->nums = (long *) cacmem(num * sizeof(long));
    cac->next = (short *) cacmem(num * sizeof(short));
    cac->prio = (short *) cacmem(num * sizeof(short));
    cac->mark = cacmem(num);
    cac->recs = cacmem(num * recl);

    /* initial lru/mru chain */

    while (i < num) {
        cac->next[i] = i + 1;
        cac->prio[i] = i - 1;
        cac->nums[i] = -1;
        cac->mark[i] = 0;
        i++;
    }
```


Listing Two (Listing continued, text begins on page 30.)

```

cac->next[num - 1] = cac->prio[0] = -1;
cac->mru = 0;
cac->lru = num - 1;

/* return cache pointer */
return cac; }

/* allocate memory with error checking */
char *cacmem(siz)
int    siz;
{
    char *c;
    c = malloc(siz);
    if (c == (char *) 0) {
        fprintf(stderr, "cacallo: Can't allocate memory.\n");
        fprintf(stderr, "Tried to get %d bytes on top of %ld bytes already\n",
            siz, cacmem);
        exit(1); }
    cacmem += siz;
    return c; }

/* number oldest record and make it the newest.  if the record was
marked for exit procesing, call external processing function.
return pointer to record */

char *cacnum(cac, num)
CACDS *cac;          /* cache header */
long   num;           /* number new MRU record */
{
    char *rec = cac->recs + (cac->lru * cac->recl);

    /* call external function */
    if (cac->mark[cac->lru] && cac->proc)
        (*(cac->proc))(cac->idnt, cac->nums[cac->lru], rec);

    /* unmark record and make it newest */

    cac->mark[cac->lru] = 0;
    cac->adds++;
    cac->nums[cac->lru] = num;
    cacnew(cac, cac->lru);

    /* return record; ready for usage */
    return rec; }

```

(continued on next page)

Announcing WKS LIBRARY

NEW!

The Lotus "Wrap-Around" for C Programs

Now you can write and read Lotus worksheets
directly from a C program!

WKS LIBRARY lets you:

- avoid time-consuming file translation steps
- control the execution of 1-2-3 from inside your application
- use Lotus as a data entry screen in your C program
- generate "live" financial statements with formulas for totals

Feature this:

- reads and writes .WKS, .WK1 and .WR1 worksheets
- writes integers, floats, strings, formulas, macros and dates using *wprintf()*
- reads using *wscanf()*, converting column contents to C variables according to format specs
- has low-level functions for manipulating individual cells
- provides Lotus control functions such as:
formats, column widths, initial cursor position, range names, cell protection
- supports Lattice, Microsoft & Turbo compilers for DOS, plus most UNIX environments

Source Code provided No Royalties on executable programs

Only \$89

(includes free
800-line support)



ORDER TODAY!

Toll-free
(800) 367-9882



Tenon Software, Inc.

1980 - 112th NE, #250, Bellevue, WA 98004
(206) 453-1914 (in Washington state)

CANADA'S SOURCE FOR C

- Canadian Sales
- Canadian Service
- Canadian Technical Support
- Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX

SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL

AGE OF REASON • DESMET • AZTEC

MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS

HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •



Call for full price list—Dealer enquiries welcome



We know our products—we use them!

SCANTEL SYSTEMS LTD.

801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

CIRCLE 139 ON READER SERVICE CARD

RAM-CACHE MANAGER

Listing Two (Listing continued, text begins on page 30.)

```
/* get pointer to oldest record without altering age */
char *cacold(cac)
CACDS *cac; /* cache header */
{
    return cac->recs + (cac->lru * cac->recl); }

/* if an exit processing routine has been defined, process all marked
records */
cacflush(cac)
CACDS *cac; /* cache header */
{
    int i;
    char *rec;

    if (!(cac->proc)) return;

    for (i = 0; i < cac->maxr; i++) if (cac->mark[i]) {
        rec = cac->recs + (i * cac->recl);
        (*(cac->proc))(cac->idnt, cac->nums[i], rec);
        cac->mark[i] = 0; }

    return; }

/* make record newest */
cacnew(cac, rec)
CACDS *cac; /* cache header */
short rec; /* record to make newest */
{
    /* if this record is already the newest, just return */
    if (rec == cac->mru) return;

    /* change prior's next */
    cac->next[cac->prio[rec]] = cac->next[rec];

    /* change next's prior - if there was no next that means this was the
lru record. change lru */
    if (cac->next[rec] != -1)
        cac->prio[cac->next[rec]] = cac->prio[rec];
    else {
        if (rec != cac->lru) {
            fprintf(stderr, "cacnew: panic\n");
            exit(1); }
        cac->lru = cac->prio[rec]; }

    /* now the record is out of the chain. stick it back in at the mru end. */
    cac->prio[cac->mru] = rec;
    cac->next[rec] = cac->mru;
    cac->prio[rec] = -1;
    cac->mru = rec;

    /* done */
    return; }

/* find record and make it newest */
char *cacfind(cac, num)
CACDS *cac; /* cache header */
long num; /* record number to look for */
{
    int i;

    for (i = 0; i < cac->maxr; i++) if (cac->nums[i] == num) {
        cac->hits++;
        cacnew(cac, i);
        return cac->recs + (i * cac->recl); }

    cac->miss++;
    return (char *) 0; }

/* mark record for external processing */
cacproc(cac, num)
CACDS *cac; /* cache header */
long num; /* record to mark */
{
    int i;

    for (i = 0; i < cac->maxr; i++) if (cac->nums[i] == num) {
        cac->mark[i] = 1;
        return; }

    return; }

/* un-mark record for external processing */
cacunpc(cac, num)
CACDS *cac; /* cache header */
long num; /* record to unmark */
{
    int i;

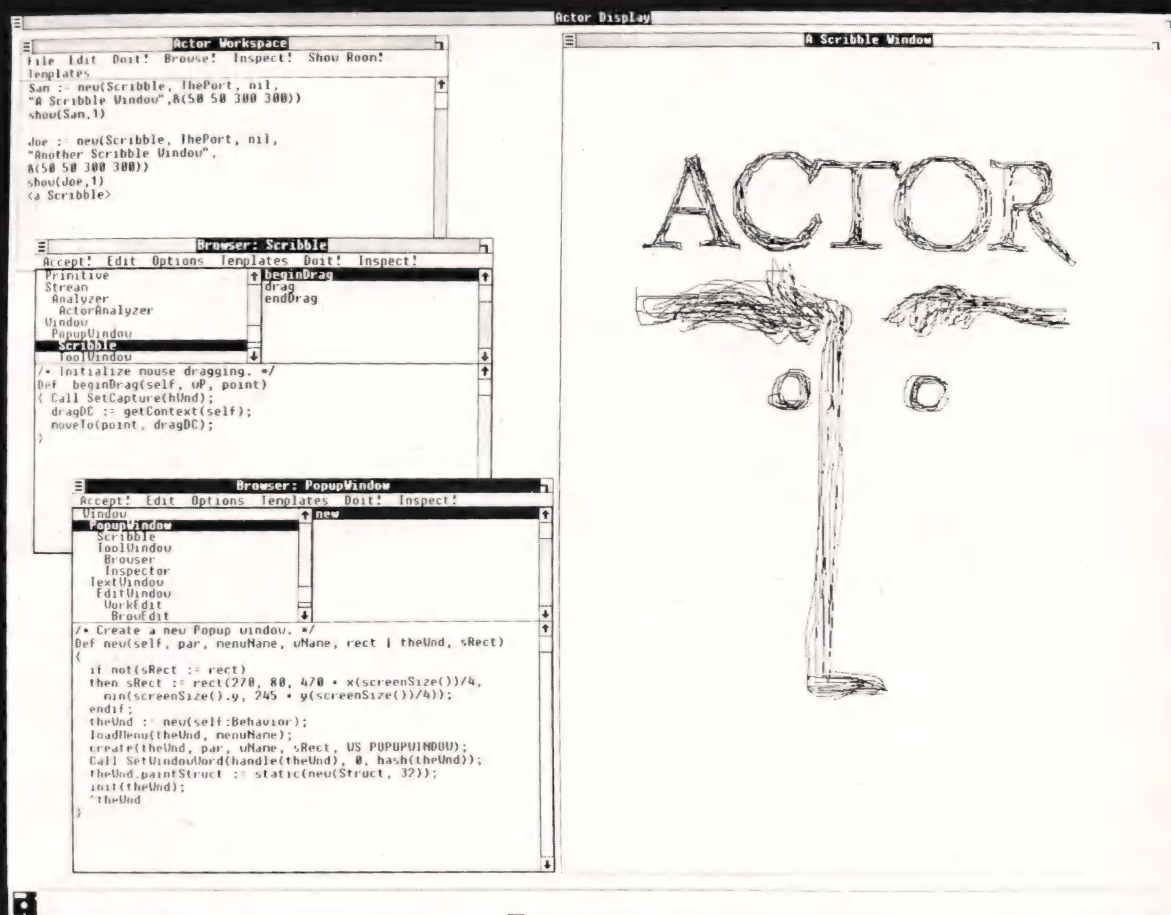
    for (i = 0; i < cac->maxr; i++) if (cac->nums[i] == num) {
        cac->mark[i] = 0;
        return; }

    return; }

/* get statistics of the cache */
```

(continued on page 67)

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor.
Call The Whitewater Group, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201



Call or write
for the latest catalog

DISCOVER PARADISE



LIST OURS	
OPERATING SYSTEMS	
MICROPORT SYSTEM V/AT (COMPLETE)	549 465
SCO XENIX SYSTEM V (COMPLETE)	1295 995
WENDIN-DOS	99 79
OTHER MICROPORT, SCO, WENDIN PRODUCTS	CALL CALL
PASCAL COMPILERS	
MARSHAL PASCAL	SPECIAL 189 149
MICROSOFT PASCAL	300 185
PASCAL-2	SPECIAL 350 299
TURBO PASCAL	NEW V. 4.0 100 65
TURBO PASCAL DEV. LIB.	NEW 395 259
BORLAND ADD-ONS	CALL CALL

FEATURED PRODUCTS

PRO-C 'C' — Source Program Generator. It produces stand-alone 'C' program exactly to your specification. Each program is fully optimized and ready to run. They are complete with documented source code and full system documentation. PRO-C gives freedom from the time-consuming work associated with Source Code development, allowing concentration on more demanding tasks such as system design and specification.
List: \$399 Special Price: \$379

CARBON COPY PLUS — Communications software, that features both Remote Control and Terminal Emulation.
Remote Control: Use your PC to observe and control a remote PC as though the two were one. Access your office PC while you are at home.
Terminal Emulation: Includes V-52, V-100, TVI-320, IBM-3101. Easy to access host computers and on-line information databases. Carbon Copy Plus will also allow you to send and receive telexes.
List: \$195 Special Price: \$139

Pascal-2 — Highly optimized Pascal compiler, with source level debugger, profiler.
List: \$350 Special Price: \$299

ADVANTAGE Disassembler — Provides immediate feedback as you work, strong results in tables on disk. Final output is ready for MS assembler. Handles COM and EXE files 8086/186/286 code and 8087/287 coprocessors.
List: \$295 Special Price: \$249

LIST OURS	
TURBO PASCAL ADD-ONS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
SCREEN DISPLAY/WINDOWS	
C-SCAPE	279 CALL
CURSES W/SOURCE CODE	250 165
GREENLEAF DATA WINDOWS	225 155
W/SOURCE CODE	395 289
HI-SCREEN XL	149 119
JVACC FORMAKER	995 449
JVACC JAM	760 669
MICROSOFT WINDOWS	99 65
MS WINDOWS DEVELOPMENT KIT	500 309
PANEL PLUS	495 395
PANEL FOR QUICKC/TURBO C	129 95
QUICKSCREEN	195 175
SCREEN ACE	SPECIAL 195 159
SCREENSTAR W/SOURCE	198 155
VITAMIN C	225 159
VC SCREEN	99 79
WINDOWS FOR DATA	295 235
VIEW MANAGER	275 199
ZVIEW	245 169

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

LIST OURS	
ALICE	95 69
DOS/BIOS & MOUSE TOOLS	75 69
FLASH-UP	89 79
METABYTE DATA ACQ. TOOLS	100 89
SCREEN SCULPTOR	125 95
SYSTEM BUILDER	150 129
IMPEX	100 89
REPORT BUILDER	130 115
TURBO ASM	99 69
T-DEBUG PLUS	60 49
TURBO ASYNCH PLUS	NEW 129 99
TURBO EXTENDER	85 65
TURBO HALO	129 99
TURBO MAGIC	99 89
TURBO MASTER	125 99
TURBO OPTIMIZER	75 65
TURBO POWER TOOLS PLUS	NEW 129 99
TURBO POWER UTILITIES	95 79
TURBO WINDOW/PASCAL	95 79

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The microcomputer software source that caters to your programming needs. Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee

Over 500 brand-name products in stock—if you don't see it, call!

We'll Match Any Nationally Advertised Price.

LIST OURS	
386 SOFTWARE	
ADVANTAGE 386 C	895 799
ADVANTAGE 386 PASCAL	895 799
MICROPORT SYSTEM V/386 (COMPLETE)	SPECIAL 799 679
MICROSOFT WINDOWS/386	195 125
PHARLAP 386/ASM/LINK	495 419
PHARLAP 386 DEBUG	195 155
SCO XENIX SYS V 386 (COMPLETE)	1495 1195
VM/386	SPECIAL 195 119
ARTIFICIAL INTELLIGENCE	
ARITY STANDARD PROLOG	95 CALL
GOLDEN COMMON LISP	250 155
MICROSOFT LISP	250 155
PC SCHEME	95 85
SMALLTALK V	99 85
TURBO PROLOG	100 65
TURBO PROLOG TOOLBOX	100 65
ASSEMBLERS/LINKERS	
ADVANTAGE DISASM.	SPECIAL 295 249
ADVANTAGE LINK	395 359
ASMLIB	149 125
EZ-ASM	70 65
MS MASM	CALL REBATE OFFER 150 95
PASM86	195 109
PLINK86PLUS	495 275
RELMS CROSS ASSEMBLERS	CALL CALL
UNWARE CROSS ASSEMBLERS	CALL CALL
VISIBLE COMPUTER 80286	100 89
BASIC	
DB/LIB	99 89
FLASH-UP	89 79
MACH 2	75 59
MS QUICKBASIC	CALL REBATE OFFER 99 65
QUICKPAK	69 59
TRUE BASIC	100 69
TURBO BASIC	100 65
DATABASE TOOLBOX	100

RAM-CACHE MANAGER

Listing Two (Listing continued, text begins on page 30.)

```

cacstat(cac, hit, mis, add)
CACDS *cac; /* cache header */
long *hit, *mis, *add; /* values to return */
{
    *hit = cac->hits;
    *mis = cac->miss;
    *add = cac->adds;
    return; }

/* free cache */

cacfree(cac)
CACDS *cac; /* cache header */
{
    cacflsh(cac);
    cacmem -= sizeof(CACDS) + (cac->maxr * sizeof(long)) +
              (cac->maxr * 2 * sizeof(short)) + (cac->maxr) +
              (cac->maxr * cac->recl);
    free(cac->recs);
    free(cac->prio);
    free(cac->next);
    free(cac->nums);
    free(cac);
    return; }

```

End Listing Two

Listing Three

```

Listing 3

#include <stdio.h>
#include "cache.h"
CACDS *cache;

main() {
    printf("Cache test routine\n");
    cache = cacallo(8, 128, (char *) 0, 1L);
    printf("Memory allocated - %ld\n", cacmem);

    while (ctest()) cprint();

    exit(0); }

int ctest() {
    int opt, rec;
    long num;

    printf("\n1-old, 2-num, 3-find, 4-proc: ");
    scanf("%d", &opt);

    switch (opt) {
        case 0: return 0;
        case 1: printf("cacold returns %lx\n", cacold(cache)); return 1;
        case 2: printf("enter record: ");
                scanf("%ld", &num);
                printf("cacnum returned %lx\n", cacnum(cache, num));
                return 1;
        case 3: printf("input record to find: ");
                scanf("%ld", &num);
                printf("cacfind returned %lx\n", cacfind(cache, num));
                return 1;
        case 4: printf("input record to process: ");
                scanf("%ld", &num);
                cacproc(cache, num);
                printf("cacproc called\n");
                return 1;
        otherwise: return 1; }
    return; }

cprint() {
    register int i;

    printf("Cache print: hits=%ld miss=%ld adds=%ld\n",
           cache->hits, cache->miss, cache->adds);

    printf("Block Numbers Next Prior Mark LRU=%d MRU=%d\n", cache->lru, cache->mr);
    printf("-----\n");
    for (i = 0; i < cache->maxr; i++)
        printf("%5d %7ld %5d %5d %4d\n", i, cache->nums[i],
              cache->next[i],
              cache->prio[i],
              cache->mark[i]);

    return; }

```

End Listings

DAN BRICKLIN'S DEMO PROGRAM ONLY \$74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."

—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."

—Soft-letter

Product of the Month

—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST \$49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!

1-800-CALL-800 x8088

Use 800-number for orders only. Questions, special shipping, etc., call 617-332-2240. No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add \$15.00. Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



SOFTWARE GARDEN, INC.

Dept. D
P.O. Box 373, Newton Highlands, MA 02161
CIRCLE 142 ON READER SERVICE CARD

Listing One (Text begins on page 38.)

BOOT.C

```

1  #include <stdio.h>
2  #include <io.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <malloc.h>
6
7  #include "loc.h"
8  #include "externs.h"
9
10
11 void create_bootstrap(seg_list, entry)
12 SEG_DESCRIPTOR *seg_list ;
13 unsigned char *entry ;
14 {
15     unsigned int    count ;
16     unsigned char  *ptr ;
17
18     SEG_DESCRIPTOR *p, *q ;
19
20     /*
21      * This function sets up a new class which contains the bootstrap
22      * code to the program entry point. The bootstrap segment is
23      * always located at physical address FFFF0H is ROMable.
24      *
25      * The bootstrap record is appended to the load module for the
26      * purpose of locate processing.
27      */
28
29     /* Traverse the linked list to the end */
30     p = seg_list ;
31     while (p->next != NULL)
32         p = p->next ;
33
34     /* Allocate the memory for the bootstrap record */
35     if ((q = (SEG_DESCRIPTOR *) malloc(sizeof (*p))) == NULL) {
36         perror(_FILE_) ;
37         exit(1) ;
38     }
39
40     /* Append the bootstrap record to the end of the list */
41     p->next = q ;
42
43     /* Initialize the bootstrap segment descriptor */
44     strcpy(q->name, "?BOOT") ;
45     strcpy(q->class, "(ABSOLUTE)") ;
46     q->vseg = 0xffff ;
47     q->pseg = 0xffff ;
48     q->offset = 0x0000 ;
49     q->len = 5 ;
50     q->inited = TRUE ;
51     q->romable = TRUE ;
52     q->symbols = 0 ;
53     q->symbol_list = NULL ;
54     q->next = NULL ;
55
56     /* Allocate RAM and build the reset vector code using a far jump */
57     ptr = malloc(q->len) ;
58     *ptr = 0xEA ;
59     *((unsigned char **) (ptr + 1)) = entry ;
60
61     /* Append the bootstrap code on tail of the load module */
62     q->position = lseek(tmp_file, 0L, SEEK_END) ;
63     count = write(tmp_file, ptr, q->len) ;
64     if (count != q->len) {
65         perror(_FILE_) ;
66         exit(1) ;
67     }
68
69     free(ptr) ;
70     return ;
71 }

```

End Listing One

Listing Two

FILES.C

```

1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <sys\types.h>
4  #include <sys\stat.h>
5  #include <io.h>
6  #include <string.h>
7
8  #include "loc.h"
9  #include "externs.h"
10
11 #define F_OPEN      O_RDONLY | O_BINARY
12 #define F_CREATE    O_CREAT | O_TRUNC | O_RDWR | O_BINARY
13
14
15 void open_file_system(input_file)
16 char *input_file ;
17 {
18     char *create_str = "Can't create %s" ;
19     char *open_str = "Can't open %s" ;
20     char errmsg[MAX_LINE] ;
21     char *filename_ext ;
22
23     /*
24      * This module is responsible for opening or creating all of the
25      * files used by this utility.
26      */
27

```

(continued on page 73)



Our software comes with something no one else can offer.

When you join the Lattice family of customers, you'll discover that your software purchase is backed by more than just an excellent warranty. It's backed by unparalleled technical support. By a total commitment to your success and satisfaction. And by Lattice's dedication to excellence in products and services.

Unlike other software manufacturers who charge you for services after you've purchased their product, Lattice offers a unique package of support programs at a price we can all live with—FREE.

Lattice Bulletin Board Service

LBBS is our 24-hour a day bulletin board system that allows you to obtain notification of new releases, general information on Lattice products, and programs for the serious user. And if you've ever experienced the frustration of having to wait a year or more for a new release (that has corrected a bug), you'll really appreciate LBBS. Because with this service, you can actually download the latest program fixes to instantly eliminate any bugs discovered after release.

Available through dealers and distributors worldwide.

Lattice Service.

Technical Support Hotline

Responsible, dependable and capable Support Representatives are only a phone call away. You will talk to a highly skilled expert who is trained to answer any questions you have relating to specific Lattice products. Remember, your complete satisfaction is our goal.

McGraw-Hill BIX™ Network

The Byte Information Exchange (BIX) Network is a dial-in conference system that connects you with a Special Interest Group of Lattice users. The nominal one-time registration fee allows you to BIX-mail your questions—via your modem—directly to Lattice. Or you can post your questions in the conference mode for Lattice or other users to answer. Once again, you have 24-hour access.

You Also Receive:

- Timely updates and exciting enhancements
- 30-day, money-back guarantee
- Lattice Works Newsletter
- Technical Bulletins
- Access to Lattice User Groups

Lattice has developed more than 50 different Microcomputer software tools that are used by programmers worldwide. We were there for every MS-DOS release. We're there now for OS/2. And we'll be there for the next generation of technical changes. But most of all, Lattice is there for you.



Lattice

Subsidiary of SAS Institute Inc.

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

CIRCLE 143 ON READER SERVICE CARD

This ad is for people who don't know where to find Smalltalk. Or why.

Today, the single most important emerging software technology is OOPS, object-oriented programming. It's destined to dramatically change the way you use your personal computer. You'll find it doing things you never expected. And by people you never suspected.

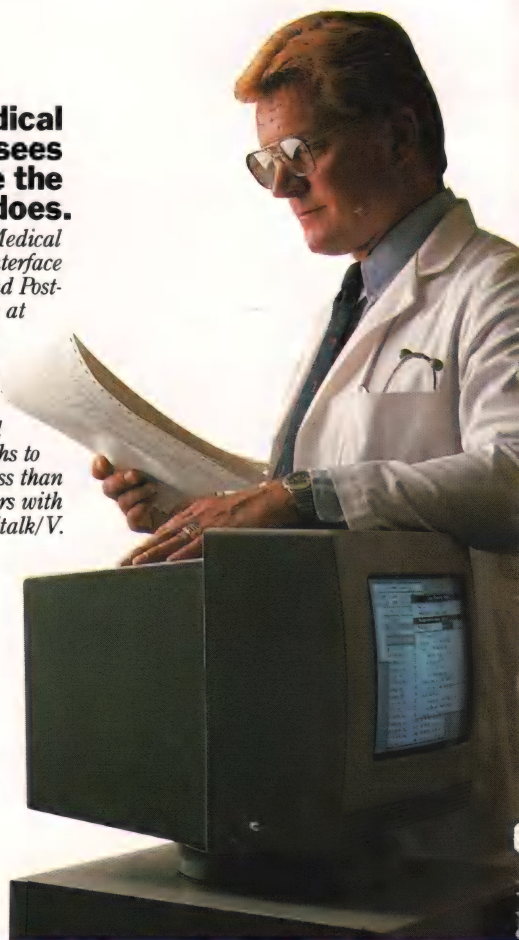
In an emergency room in Vancouver, it's saving lives through animation.

What if a medical textbook could come to life? What if it could show the effects emergency treatment might have on patients? And do it all through moving pictures? These thoughts led Folkstone Design, Edge Training & Consulting, and Inform Software in Vancouver, B.C., to create the first animated, interactive textbook for emergency room technicians and in-training paramedics. They found Smalltalk/V could easily facilitate a combination of text, color graphics and animation to illustrate various physical processes and the results of medical intervention.



At the UCLA Medical Center, it sees patients before the doctor does.

Mike McCoy, M.D., at the UCLA Medical Center, found that he could easily interface Smalltalk/V with dBASEIII and PostScript. His application, now in use at the Clinic, turns a functional status questionnaire on each new patient into a laser printed, advisory analysis for the doctor to review prior to seeing the patient. A program like this would normally take a specialist months to produce. It took Dr. McCoy less than 100 hours with Smalltalk/V.



It's working on Florida's freeways.

Running on IBM's new PS/2, a Smalltalk/V application developed by Greiner Engineering's Mike Rice, lets highway engineers create highly sophisticated graphic analyses of any proposed reconstruction. So now, instead of having to deal with a gridlock of Federal and State regulations, engineering specifications and endless calculations, an engineer can quickly explore alternative design strategies using a mouse, windows and VGA color graphics.

Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected. dBASEIII, PostScript and PS/2 are trademarks of Ashton-Tate, Adobe Systems and International Business Machines Corporation respectively.



It's tracking white-tail deer on the Barrier Islands of Georgia.

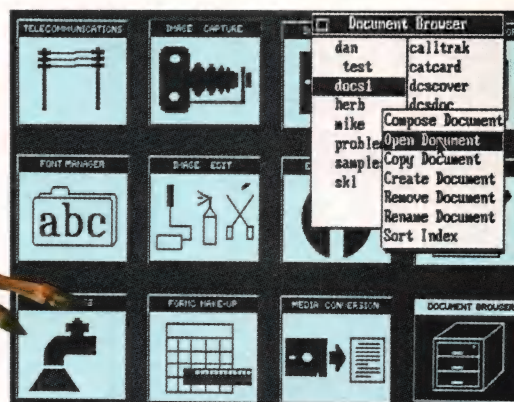
Dr. Lee Graham, a National Park Service ecologist chose Smalltalk/V to write an application to help manage the white-tail deer population on the Barrier Islands of Georgia. Dr. Graham found that Smalltalk/V, with its visual interface and class structure, is a perfect tool to graphically simulate the complex, ecological interactions of natural systems.



You can find it in space.

On a project commissioned by NASA, Dr. Christine Mitchell at the Georgia Institute of Technology, chose to use Smalltalk/V as an integral part of a new man-machine interface. The application, written in Smalltalk, continually monitors the commands of the Satellite Network Operator, the state-of-the-network and the overall mission plans.

To NASA, Smalltalk/V means real-time. Real OOPS. Real results.



It's making headlines in Arizona.

When Digital Composition Systems sat down to build an electronic typesetting system, they had three major requirements. It had to have the most advanced user interface. It had to be fast. And, it had to be able to turn untrained personnel into high quality typographers. Of all the languages in the world, they chose Smalltalk/V. The result is the Signature Series, recognized and reviewed by The Seybold Report. It's now marketed by Digital Composition Systems and one of the largest digital typesetting firms in the world, Varityper AM International.

What thousands of people have found is OOPS.

Object-Oriented Programming (OOPS) is programming by defining objects, their inter-relationships and their behavior. Objects can represent both real-world entities like people, places, or things. They can also represent useful abstractions such as stacks, sets and rectangles.

OOPS models the way you think and the way things really are. It lets you solve problems by breaking them down into easily handled sub-problems and their inter-relationships. The solutions you come up with can be re-used to solve new problems. Ultimately, OOPS makes programming a simple,

logical process of building on the work of others.

Why thousands more are finding their way to Smalltalk/V.

First of all, Smalltalk/V makes OOPS easy.

It's also fast. In fact, it's the fastest OOPS programming available on a PC.

And it's easy to learn. It comes complete with a tutorial that's the best introduction to OOPS available.

Smalltalk/V also has a few other features worth noting. Like a user-extendable, open ended environment. Source code with browser windows for easy access and modification. A huge toolkit of classes and objects for building a variety of applications. A sophisticated source-level debugger. Object-oriented Prolog integrated with the Smalltalk environment. And bit-mapped graphics with bit and form editors, just to name a few.

Then, there's its unbelievable price of only \$99.95. (Optional application packs at \$49.95 include Communications, EGA/VGA Color and Goodies.)

And it has a 60 day, money-back guarantee.

With all this to offer, it probably won't come as a surprise to you that more people are solving more problems with Smalltalk/V than any other OOPS.

See your nearest dealer today for your own Smalltalk/V. Or, order it direct with MasterCard or Visa at (800) 922-8255.

Or, write to Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045. Then discover all the great things you can do with your PC and Smalltalk/V.

Smalltalk/V

digitalk inc.



*Now that you've found us, write us. Tell us some of the great things you're doing with Smalltalk/V. You could be in our next ad.



QNX vs. OS/2 UNIX

QNX: Bend it, shape it, any way you want it.

ARCHITECTURE If the micro world were not so varied, QNX would not be so successful. After all, it is the operating system which enhances or limits the potential capabilities of applications. QNX owes its success (over 30,000 systems sold since 1982) to the tremendous power and flexibility provided by its modular architecture.

Based on message-passing, QNX is radically more innovative than UNIX or OS/2. Written by a small team of dedicated designers, it provides a fully integrated multi-user, multi-tasking, networked operating system in a lean 148K. By comparison, both OS/2 and UNIX, written by many hands, are huge and cumbersome. Both are examples of a monolithic operating system design fashionable over 20 years ago.

MULTI-USER OS/2 is multi-tasking but NOT multi-user. For OS/2, this inherent deficiency is a serious handicap for ter-

minal and remote access. QNX is both multi-tasking AND multi-user, allowing up to 16 terminals and modems to connect to any computer.

INTEGRATED NETWORKING Neither UNIX nor OS/2 can provide integrated networking. With truly distributed processing and resource sharing, QNX makes all resources (processors, disks, printers and modems anywhere on the network) available to any user. Systems may be single computers, or, by simply adding micros without changes to user software, they can grow to large transparent multi-processor environments. QNX is the main-frame you build micro by micro.

PC's, AT's and PS/2's OS/2 and UNIX severely restrict hardware that can be used: you must replace all your PC's with AT's. In contrast, QNX runs superbly on PC's and literally soars on AT's and PS/2's. You can

run your unmodified QNX applications on any mix of machines, either standalone or in a QNX local area network, in real mode on PC's or in protected mode on AT's. Only QNX lets you run multi-user/multi-tasking with networking on all classes of machines.

REAL TIME QNX real-time performance leaves both OS/2 and UNIX wallowing at the gate. In fact, QNX is in use at thousands of real-time sites, right now.

DOS SUPPORT QNX allows you to run PC-DOS applications as single-user tasks, for both PC's and AT's in real or protected mode. With OS/2, 128K of the DOS memory is consumed to enable this facility. Within QNX protected mode, a full 640K can be used for PC-DOS.

ANY WAY YOU WANT IT QNX has the power and flexibility you need. Call for details and a demo disk.

THE ONLY MULTI-USER, MULTI-TASKING, NETWORKING, REAL-TIME OPERATING SYSTEM FOR THE IBM PC, AT, PS/2, THE HP VECTRA, AND COMPATIBLES.

Multi-User	10 (16) serial terminals per PC (AT).	C Compiler	Standard Kernighan and Ritchie.
Multi-Tasking	40 (64) tasks per PC (AT).	Flexibility	Single PC, networked PC's, single PC with terminals, networked PC's with terminals. No central servers. Full sharing of disks, devices and CPU's.
Networking	2.5 Megabit token passing. 255 PC's and/or AT's per network. 10,000 tasks per network. Thousands of users per network.	PC-DOS	PC-DOS runs as a QNX task.
Real Time	2,800 task switches/sec (AT).	Cost	From US \$450. Runtime pricing available.
Message Passing	Fast intertask communication between tasks on any machine.		

For further information or a free demonstration diskette, please telephone (613) 591-0931.

Quantum Software Systems Ltd. • Kanata South Business Park • 175 Terrence Matthews Crescent • Kanata, Ontario, Canada • K2M 1W8

UNIX is a registered trademark of AT & T Bell Labs. IBM, PC, AT, XT and PS/2, PC-DOS and OS/2 are trademarks of International Business Machines. HP and Vectra are registered trademarks of Hewlett-Packard Company.

CIRCLE 145 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Two

(Listing continued, text begins on page 38.)

```

28  /* Perform all the filename processing */
29  strcpy(module_name, strupr(input_file));
30  strcpy(exe_fname, module_name);
31  strcat(strcpy(map_fname, exe_fname), ".MAP");
32
33  if ((config == FALSE) || (strlen(config_fname) == 0))
34    strcat(strcpy(config_fname, exe_fname), ".CFG");
35  else
36    strupr(config_fname);
37
38  if ((hex_name == FALSE) || (strlen(abs_fname) == 0))
39    strcat(strcpy(abs_fname, exe_fname), ".HEX");
40  else
41    strupr(abs_fname);
42
43  strcat(strcpy(print_fname, exe_fname), ".LOC");
44  strcat(exe_fname, ".EXE");
45
46  /* Create the temporary file used for segment fixups and
                                location */
47  strcpy(tmp_fname, "LOCATE.$$$");
48  tmp_file = open(tmp_fname, F_CREATE, S_IWRITE);
49  if (tmp_file == -1) {
50    sprintf(errmsg, create_str, tmp_fname);
51    perror(errmsg);
52    exit(1);
53  }
54
55  /* Create the absolute output file */
56  abs_file = open(abs_fname, F_CREATE, S_IWRITE);
57  if (abs_file == -1) {
58    sprintf(errmsg, create_str, abs_fname);
59    perror(errmsg);
60    exit(1);
61  }
62
63  /* Open the .EXE file */
64  exe_file = open(exe_fname, F_OPEN);
65  if (exe_file == -1) {
66    sprintf(errmsg, open_str, exe_fname);
67    perror(errmsg);
68    exit(1);
69  }
70
71  /* Create the locate map output file */
72  print_file = fopen(print_fname, "wt");
73  if (pPrint file == NULL) {
74    sprintf(errmsg, open_str, print_fname);
75    perror(errmsg);
76    exit(1);
77  }
78
79  /* Open the configuration file for reading */
80  config_file = fopen(config_fname, "rt");
81  if (config_file == NULL) {
82    sprintf(errmsg, open_str, config_fname);
83    perror(errmsg);
84    exit(1);
85  }
86
87  /* Open the linker map file for reading */
88  map_file = fopen(map_fname, "rt");
89  if (map_file == NULL) {
90    sprintf(errmsg, open_str, map_fname);
91    perror(errmsg);
92    exit(1);
93  }
94
95  return;
96 }
97
98 void close_file_system()
99 {
100  char  errmsg[MAX_LINE];
101  char  *close_str = "Unable to close %s";
102  char  *delete_str = "Unable to delete %s";
103
104  /*
105   This function is responsible for shutting down the file system
106   and cleaning up the temporary files. All files opened for
107   reading are closed and all files open for writing are closed
108   (normal exit) and possibly deleted (control-C abort event).
109  */
110
111  /* Close the link map */
112  if (fclose(map_file) != 0) {
113    sprintf(errmsg, close_str, map_fname);
114    perror(errmsg);
115  }
116
117  /* Close the locate configuration file */
118  if (fclose(config_file) != 0) {
119    sprintf(errmsg, close_str, config_fname);
120    perror(errmsg);
121  }
122
123  /* Close the .EXE file */
124  if (close(exe_file) != -1) {
125    sprintf(errmsg, close_str, exe_fname);
126    perror(errmsg);
127  }
128 }

```

(continued on next page)

TOTAL CONTROL with LMI FORTH™



For Programming Professionals:
an expanding family of
compatible, high-performance,
Forth-83 Standard compilers
for microcomputers

For Development:

Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: (213) 306-7412

Overseas Distributors.

Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

TRUE MULTITASKING

With

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
 - * Intertask message communication. Send/receive/check message present on 64 message queues.
 - * Task control by means of semaphores. Get/release/check semaphores.
 - * Change priority-256 priority levels.
 - * Suspend task for specified interval.
 - * Spawn and terminate external and internal tasks.
 - * Disable/enable multitasking.
 - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **\$24.95** OR
\$99.95
with source code

Outside USA add \$5.00 shipping and handling.

Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE 147 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Two (Listing continued, text begins on page 38.)

```

129
130 /* Close the locate map */
131 if (fclose(print_file) != 0) {
132     sprintf(errmsg, close_str, print_fname);
133     perror(errmsg);
134 }
135
136 /* Close the absolute or hex object module */
137 if (close(abs_file) == -1) {
138     sprintf(errmsg, close_str, abs_fname);
139     perror(errmsg);
140 }
141
142 if (user_abort == TRUE) {
143     /* Delete the locate map */
144     if (remove(print_fname) == -1) {
145         sprintf(errmsg, delete_str, print_fname);
146         perror(errmsg);
147     }
148
149     /* Delete the object file */
150     if (remove(abs_fname) == -1) {
151         sprintf(errmsg, delete_str, abs_fname);
152         perror(errmsg);
153     }
154 }
155
156 /* Close and then delete the temporary file */
157 if (close(tmp_file) == -1) {
158     sprintf(errmsg, close_str, tmp_fname);
159     perror(errmsg);
160 }
161
162 if (remove(tmp_fname) == -1) {
163     sprintf(errmsg, delete_str, tmp_fname);
164     perror(errmsg);
165 }
166
167 return;
168 }
    
```

End Listing Two

Listing Three

LOADEXE.C

```

1  #include <stdio.h>
2  #include <io.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <malloc.h>
6  #include <dos.h>
7
8  #include "loc.h"
9  #include "externs.h"
10
11
12 char *warn_str = "Warning: Unable to locate virtual segment %04X\n";
13
14 char *load_exe_file()
15 {
16     char buf[128];
17     int count, i;
18     long seek_pos;
19     unsigned int module_size, pseg, *reloc_ptr, segment;
20     unsigned int read_size, mem_size;
21     char *load_addr, *entry, *str;
22
23     EXE_HEADER header;
24     SEG_DESCRIPTOR *p;
25
26     /*
27      * This function reads in the .EXE file and performs the fixup of any
28      * segment references.
29      */
30
31     /* Read in the .EXE file header information */
32     count = read(exe_file, (char *) &header, sizeof(header));
33     if (count != sizeof(header)) {
34         perror(_FILE_);
35         exit(1);
36     }
37
38     /* Exit if not a valid .EXE file */
39     if (header.signature != 0x5A4D) {
40         perror("Not an .EXE file");
41         exit(1);
42     }
43
44     /* Seek to the start of the load module */
45     if (lseek(exe_file, (long) header.header_size * 16, SEEK_SET) == -1L) {
46         perror(_FILE_);
47         exit(1);
48     }
49
50     /* Compute how much memory can be allocated for reads */
51     mem_size = 32 * 1024;
52     if (mem_size > _memavl())
53         mem_size = _memavl();
54
55     /* Allocate the memory */
56     if ((load_addr = malloc(mem_size)) == NULL) {
    
```

(continued on page 76)

UNLEASH YOUR 80386!

Your 80386-based PC runs at least twice as fast as your old AT. This is good, but not great. The products described below will unleash the true potential of your 80386, giving you 4 to 16 times the power of your old AT. These new MicroWay products include a family of 80386 native code compilers and the mW1167 numeric coprocessor.

Examples of the increases in capacity and performance include:

- Programs compiled with MicroWay

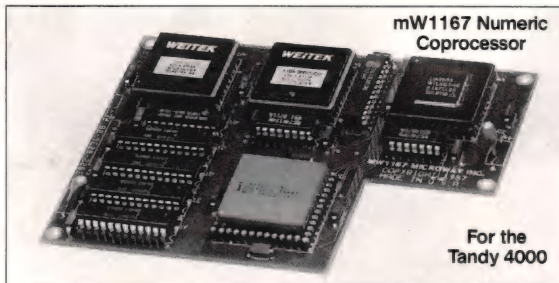
NDP Fortran-386 execute 2 to 8 times faster than those compiled with existing 16-bit Fortrans. NDP Fortran-386 can also address up to 4 gigabytes of memory instead of the standard 640 kbytes. MicroWay's NDP compilers and the programs they generate run on MS-DOS or Unix V.

- NDP Fortran-386 generates code for the 80287, 80387 or MicroWay's mW1167. The mW1167 has a floating point throughput exceeding 2.5 mega-

flops, which is 4 to 5 times the throughput of an 80387 and is comparable to the speed achieved by the VAX 8600.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341



MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: MicroPort Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP Fortran-386™\$595
NDP C-386™\$595

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000 and MicroWay Number Smasher 386 (Jan. '88). It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 3 to 5 times faster than an 80387\$1495

Monoputer™ - The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Four or more Transputers can be easily linked together to form a Quadputer. A single T800 is comparable in speed with an mW1167-equipped 80386. The compilers to drive one or more Monoputers include Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20¹\$1495
Monoputer T800-20¹\$1995
Biputer™ T800/T414²\$4995
Quadputer™ T414-20²\$6995

¹Includes Occam ²Includes TDS

80287 ACCELERATORS

287Turbo-10\$450
287Turbo-12\$550
287TurboPlus-12\$629

80386 Multi-User Solutions

AT8™ - This intelligent serial controller is designed to handle 8 users (16 with two boards) in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems\$1299

MicroPort Unix 5.3 is a port of the new Unix 5.3 to the 80386. MicroWay NDP-386 compilers currently run on this version of UNIX.

MicroPort Unix 5.3from \$399

PC-MOS-386™ is an 80386 operating environment that turns an AT with an AT8 into an MS-DOS multi-user system. The system makes it possible to run applications such as Lotus 1-2-3 on terminals. The operating system also has a Phar Lap compatibility mode that runs programs developed with the Phar Lap versions of MicroWay's compilersfrom \$199

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers. **Phar Lap Tools**\$399

MATH COPROCESSORS

80387-16 16 MHz\$495
80287-10 10 MHz\$349
80287-8 8 MHz\$259
80287-6 6 MHz\$179
8087-2 8 MHz\$154
8087 5 MHz\$99

MicroWay

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466

Announcing . . .

The



Programmer's Power Pack

Now you can reach 100,000 programmers, consultants, and systems integrators with your postcard ad. The *Programmer's Power Pack* card deck targets this elite audience, including subscribers to *Dr. Dobb's Journal of Software Tools*, for only a little over a penny a contact!

The Programmer's Power Pack Card Deck

Next Mailing Date: December 28, 1987
Reservation Closing: November 23, 1987

For advertising rates, card specifications, and to reserve your space, contact:

Ann Roskey
 Northeast Account Manager

415-366-3600

Stephen Nestel
 National Account Manager

415-521-4133

DOS LOCATE UTILITY

Listing Three (Listing continued, text begins on page 38.)

```

57     perror(_FILE_);
58     exit(1);
59 }
60
61 while (1) {
62     /* Read in a segment of the load module */
63     read_size = read(exe_file, load_addr, mem_size);
64     if (read_size == 0) {
65         free(load_addr);
66         break;
67     }
68
69     /* Write it back out to the temporary file */
70     count = write(tmp_file, load_addr, read_size);
71     if (count != read_size) {
72         perror(_FILE_);
73         exit(1);
74     }
75 }
76
77 /* Find the relocation list */
78 if (lseek(exe_file, (long) header.first_reloc_item, SEEK_SET) == -1L) {
79     perror(_FILE_);
80     exit(1);
81 }
82
83 /* Perform the segment fixups on the temporary file */
84 for (i = 0; i < header.reloc_items; i++) {
85     /* Read in a relocation item */
86     count = read(exe_file, (char *) &reloc_ptr, sizeof(reloc_ptr));
87     if (count != sizeof(reloc_ptr)) {
88         perror(_FILE_);
89         exit(1);
90     }
91
92     /* Compute the position of the fixup in the temporary file */
93     seek_pos = (long) FP_SEG(reloc_ptr);
94     seek_pos = seek_pos * 16 + FP_OFF(reloc_ptr);
95     if (lseek(tmp_file, seek_pos, SEEK_SET) == -1L) {
96         perror(_FILE_);
97         exit(1);
98     }
99
100    /* Read in the virtual segment from the fixup */
101    count = read(tmp_file, (char *) &segment, sizeof(segment));
102    if (count != sizeof(segment)) {
103        perror(_FILE_);
104        exit(1);
105    }
106
107    /* Perform the location */
108    if (locate_virtual_segment(segment, &pseg) == ERROR)
109        fprintf(stderr, warn_str, segment);
110
111    segment = pseg;
112
113    /* Re-seek back to the fixup */
114    if (lseek(tmp_file, seek_pos, SEEK_SET) == -1L) {
115        perror(_FILE_);
116        exit(1);
117    }
118
119    /* Write the physical segment number to the fixup */
120    count = write(tmp_file, (char *) &segment, sizeof(segment));
121    if (count != sizeof(segment)) {
122        perror(_FILE_);
123        exit(1);
124    }
125 }
126
127 /* Process the program entry point */
128 if (locate_virtual_segment(header.code_seg_disp, &pseg) == ERROR)
129     fprintf(stderr, "Warning: Unable to locate entry point\n");
130
131 FP_SEG(entry) = pseg;
132 FP_OFF(entry) = header.initial_pc;
133
134 return entry;
135 }

```

End Listing Three

Listing Four

LOCATE.C

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <signal.h>
5
6  #include "loc.h"
7  #include "globals.h"
8  #include "externs.h"
9
10 /*
11  LOCATE *** MS-DOS ROM Utility
12  Copyright (C) 1987 Rick Naro. All rights reserved.
13  */
14
15 int main(argc, argv)
16 int argc;

```

(continued on page 78)

PANEL[®] Plus



Advanced Screen Manager

Building an interactive application in C? **PANEL Plus** provides the features you need for professional program development:

PRODUCTIVITY

The **PANEL Plus** interactive screen design tools are the fastest way to get your application screens set out and tested. Just type prompts on the screen, mark out entry fields, define display and entry attributes, help boxes, borders, pop-up areas. Fields can be edited, moved and resized, and validation details entered – with the screen displayed so that you can see the effect of your changes.

Then **PANEL Plus** saves your work to disk, from where you can either load the design directly into your application program, or for better control, automatically generate C data structures, field areas, and header files which are compiled and linked into your program.

QUALITY

PANEL Plus screens can include all the features demanded by today's applications. Several different menu types are provided, including highlighted bars with help lines. Easy-to-use library functions support pop-up fields, horizontal and vertical scrolling in a field, and validation exits for supplied or custom data checking functions. Text functions can also be carried out in graphics mode using a supported graphics function library.

EASE OF USE

Although the library contains over 150 functions, it is logically organised so that most programs will only need to use a small subset. Users of Roundhill's **PANEL** package will find that the old calls have been emulated (and all screens are compatible). New, expanded documentation is provided with examples of all the main function calls. **PANEL Plus** includes full library source, with variant files for all supported systems, and no royalties are payable for the use of **PANEL Plus** libraries when linked into user applications.

PORTABILITY

PANEL Plus is designed to allow your programs to be ported to just about any environment where you can find a C compiler. Every version of **PANEL Plus** includes source modules for interfacing to all of the following environments: MS-DOS/PC-DOS (portable and memory-mapped), OS/2 protected mode, Amiga Intuition, Unix (with and without *termcap* or *termio*), Xenix, DOS-J (including Asian

New !

Roundhill announces two screen tools for use with the exciting new C compilers from Borland and Microsoft:

PANEL/TC – \$129.00

For use with Borland's Turbo C

PANEL/QC – \$129.00

For use with Microsoft's Quick C

Each package is configured for use on an IBM PC or compatible system, and screens can be designed and built into your programs while running in the special development environment provided with the compiler.

All the **PANEL Plus** library functions are supported, and source code for every validation function is provided so that you can customise the entry checking to suit your application. When you need to move your programs to other environments, or need the full library source for other reasons, upgrades to **PANEL Plus** are available.

PANEL/QC can be run in any of the graphics modes supported by Quick C, and also interfaces to Microsoft C V5.

16-bit character editing), and VAX/VMS. Graphics libraries supported include MetaWindow, HALO, Essential Graphics, and Microsoft C V5. The Microsoft mouse can be used in PC versions.

PANEL Plus for MS-DOS, with full library source, is priced at \$495.00. Versions are available for the Manx Aztec, Borland, Lattice, MetaWare, Microsoft, and Wizard C compilers. Please call for prices of **PANEL Plus** for Xenix and Unix systems, and for VAX/VMS. Existing registered users of **PANEL** will receive a credit against the **PANEL Plus** license fee.

Roundhill Computer Systems Limited
PO Box 8107, Englewood NJ 07631

(201) 569 2265

Roundhill Computer Systems Limited
PO Box 14 Marlborough SN8 1LR England

(0672) 54675

Telex (UK): 444453 AWARE G
Fax (UK): (0672) 54436

BIX: join roundhill

Roundhill Computer Systems

Listing Four (Listing continued, text begins on page 38.)

```

17 char *argv[];
18 {
19     char *s, *input_file ;
20     unsigned char *entry_point ;
21     int i ;
22
23     /*
24      * This is the root module of the locate utility and it controls the
25      * sequencing of the entire location process.
26      */
27
28     /* Install a Control-C interrupt handler */
29     if (signal(SIGINT, break_handler) == (int(*)()) -1) {
30         fprintf(stderr, "Failure to install break handler\n");
31         abort();
32     }
33
34     /* Build a command line string using argv[0] through argv[argc-1] */
35     command_line[0] = '\0';
36     for (i = 0; i < argc; i++) {
37         strcat(command_line, argv[i]);
38         strcat(command_line, " ");
39     }
40
41     /* Test if the user needs help in running this utility */
42     if (argc == 1)
43         help = TRUE;
44
45     config_fname[0] = abs_fname[0] = print_fname[0] = '\0';
46
47     /* Process each argument in sequence until all are processed */
48     while (--argc > 0 && (++argv)[0] != '-') {
49         for (s = argv[0] + 1; *s != '\0'; s++) {
50             switch (*s) {
51                 case 'b':
52                     boot_rec = TRUE;
53                     break;
54
55                 case 'c':
56                     config = TRUE;
57                     if (*++s)
58                         strcpy(config_fname, s);
59                     *s-- = '\0';
60                     break;
61
62                 case 'h':
63                     hex_name = TRUE;
64                     if (*++s)
65                         strcpy(abs_fname, s);
66                     *s-- = '\0';
67                     break;
68
69                 default:
70                     help = TRUE;
71                     argc = 0;
72                     break;
73             }
74         }
75     }
76     input_file = argv[0];
77
78     if (help == TRUE) {
79         fprintf(stderr, "\nUsage is:\n\n");
80         fprintf(stderr, "\tlocate switches.exe file\n\n");
81         fprintf(stderr, "The valid switches are:\n\n");
82         fprintf(stderr, "\t\t-l4s create bootstrap record\n", "-b");
83         fprintf(stderr, "\t\t-l4s configuration filename\n", "-c[name]");
84         fprintf(stderr, "\t\t-l4s hex filename\n", "-h[name]");
85         exit(1);
86     }
87
88     fprintf(stderr, "MS-DOS Locate Utility - Version 1.0\n");
89     fprintf(stderr, "Copyright (C) 1987 Rick Naro. ");
90     fprintf(stderr, "All rights reserved\n\n");
91
92     /* Open and create the files used in the location process */
93     open_file_system(input_file);
94
95     /* Install the routine to shutdown the utility gracefully in the
96      * event of an error. */
97     onexit(close_file_system);
98
99     /* Build the segment descriptor list using the link map */
100    seg_list = build_seg_list();
101
102    /* Process the locate configuration file */
103    if (process_locate_file(seg_list) == ERROR) {
104        fprintf(stderr, "Error(s) reading the locate map\n");
105        exit(1);
106    }
107
108    /* Convert any public symbols to their new physical addresses */
109    read_symbol_table(seg_list);
110
111    /* Read the load module and perform the segment fixups */
112    entry_point = load_exe_file();
113
114    /* Add a bootstrap record if enabled on the command line */
115    if (boot_rec == TRUE)
116        create_bootstrap(seg_list, entry_point);
117
118    /* Output the load module in the specified format */
119    output_hex_OMF(abs_file, seg_list, entry_point);
120
121    /* Make the locate map containing the new segment assignments */

```



```

122     print_statistics(map_fname, print_fname, command_line, exe_fname, \
123                     abs_fname, config_fname, entry_point) ;
124
125     exit(0) ;
126 }
127
128 void break_handler()
129 {
130     /*
131      * The break handler is provided to catch Ctrl-C interrupts from the
132      * user and perform a shutdown of the program in a graceful manner.
133      */
134     /* Set the user abort flag for the file system close routine */
135     user_abort = TRUE ;
136     exit(1) ;
137 }
138
139
140

```

End Listing Four

Listing Five

MISC.C

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <malloc.h>
4  #include <dos.h>
5  #include <errno.h>
6
7  #include "loc.h"
8  #include "externs.h"
9
10 extern int  errno ;
11
12 char *get_mem(size)
13 unsigned long size ;
14 {
15     union REGS regs ;
16     char *p ;
17
18     /*
19      * This function is a substitute for allocation of huge arrays. It
20      * uses DOS system calls to directly allocate up to 64K for a memory
21      * block.
22      */
23

```

(continued on next page)

Make your C language
programs memory resident
with

DMS RESIDENT-C

Lattice 3.0

Microsoft 4.0

"hot-key"

enable

\$79.95

\$149.95
w/source



Make your assembler
programs memory resident
with

DMS RESIDENT-ASM

"hot-key"

enable

\$79.95

\$149.95
w/source



American Software International
P.O. Box 523
Windsor, Ct 06095-9998
(203) 688-5054

CIRCLE 150 ON READER SERVICE CARD

Cogent Prolog

- o Fast, Compact, COMPILED Prolog
- o IBM PC and Compatibles
- o Clocksin & Mellish Standard, Plus
 - Windows
 - Strings
 - Floating Point
 - Modules
- o Window Based Development System
- o Context Sensitive Help
- o User Specifiable Error Handling
- o Interface to "C" and Assembler
- o Can Produce .EXE Files - No Royalties

\$200

VISA - MasterCard - Check
30 Day Money Back Guarantee

Cogent Software, Ltd.

21 William J. Heights
Framingham, MA 01701
(617) 875-6553

CIRCLE 151 ON READER SERVICE CARD

goodbye dBase!



dBASE Programmers

**You need it!
You can handle it!
dB2c is here now!**

dB2c Offers:

- Version 2.0 complete with Translator and File Handlers.
- Extensive implementation of dBASE III+ with over 200 functions and commands in C source code.
- Contains our own File Handlers plus interfaces for Lattice's DBC and Faircom's c-tree.
- Supports screen I/O with ANSI.SYS or fast assembler routines.
- Support for Microsoft, Lattice and Turbo C compilers.
- Tutor features of translation combined with familiar syntax of the library eases the transition to 'C'.
- One version supports MS-DOS, Xenix, Unix, OS-9 and Concurrent DOS.

**are you
ready?**

**dB2c
Toolkit \$299**



DAVID J.
MARSH

Call or Write:

SOFTWARE
CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

DOS LOCATE UTILITY

Listing Five (Listing continued, text begins on page 38.)

```

24  regs.x.bx = (unsigned int) (size / 16 + 1) ;
25  regs.h.ah = 0x48 ;
26  FP_SEG(p) = intdos(&regs, &regs) ;
27  if (regs.x.cflag) {
28      errno = ENOMEM ;
29      p = NULL ;
30  }
31  else
32      FP_OFF(p) = 0 ;
33
34  return p ;
35  }
36
37
38  void free_mem(p)
39  char *p ;
40  {
41      union REGS regs ;
42      struct SREGS sregs ;
43
44      /*
45       * This function is the complement of get_mem() in that it releases
46       * any memory previously acquired with get_mem().
47       */
48
49      sregs.es = FP_SEG(p) ;
50      regs.h.ah = 0x49 ;
51      intdosx(&regs, &regs, &sregs) ;
52
53      return ;
54  }
55
56
57  int assign_physical_segment(class, seg)
58  char *class ;
59  unsigned int seg ;
60  {
61      int error = ERROR ;
62      SEG_DESCRIPTOR *p ;
63
64      /*
65       * This function assigns the specified class name the physical
66       * segment number. The first segment within a named class will
67       * have an offset of zero. All other segments have a segment
68       * offset relative to the first segment in the class.
69       */
70
71      p = seg_list ;
72      while (p != NULL) {
73          if (strcmp(p->class, class) == 0) {
74              p->pseg += seg ;
75              p->initd = TRUE ;
76              error = OK ;
77          }
78          p = p->next ;
79      }
80      return error ;
81  }
82
83
84  int get_next_segment(pclass, cclass, seg)
85  char *pclass ;
86  char *cclass ;
87  unsigned int *seg ;
88  {
89      int error = ERROR ;
90      BOOLEAN found = FALSE ;
91      SEG_DESCRIPTOR *p, *q, *last ;
92
93      /*
94       * This function returns the next physical segment address
95       * available for use by CCLASS (current class) after PCLASS
96       * (previous class). A typical use is to force the concatenation
97       * of independent classes.
98       */
99
100     /* Search the class list for the occurrence of the CCLASS */
101     p = q = seg_list ;
102     while (q != NULL) {
103         if (strcmp(q->class, cclass) == 0) {
104             found = TRUE ;
105             break ;
106         }
107         q = q->next ;
108     }
109
110     if (found == FALSE)
111         return ERROR ; /* Error if it can't be found */
112
113     /* Search for PCLASS and then to the end of PCLASS. */
114     while (p != NULL) {
115         if (strcmp(p->class, pclass) == 0) {
116             last = p ;
117             while (strcmp(p->class, pclass) == 0) {
118                 last = p ;
119                 p = p->next ;
120             }
121         }
122
123         /* Return the next available segment and adjust the segment
124          * value for an overflow if necessary. */

```

CIRCLE 152 ON READER SERVICE CARD


```

125     *seg = last->pseg + last->len / 16 ;
126     if (q->offset < (last->len + last->offset) % 16)
127         *seg += 1 ;
128
129     return OK ;
130 }
131 p = p->next ;
132 }
133 return ERROR ;
134 }
135
136
137 int dup_class(old_class, new_class)
138 char *old_class ;
139 char *new_class ;
140 {
141     int error = ERROR ;
142     SEG_DESCRIPTOR *p, *q, *prev, *head ;
143
144     /*
145      * Copies the contents of the OLD_CLASS entry to the newly created
146      * NEW_CLASS entry.
147      */
148
149     p = seg_list ;
150     while (p != NULL) {
151         if (strcmp(p->class, old_class) == 0) {
152             prev = head = NULL ;
153             while (p != NULL) {
154                 if (strcmp(p->class, old_class) != 0)
155                     break ;
156
157                 /* Create the new segment descriptor */
158                 if ((q = (SEG_DESCRIPTOR *) malloc(sizeof (*q))) == NULL) {
159                     perror(FILE) ;
160                     exit(1) ;
161                 }
162
163                 if (prev == NULL)
164                     head = q ;
165                 else
166                     prev->next = q ;
167
168                 /* Copy the contents and add the new entry to the list */
169                 *q = *p ;
170                 strcpy(q->class, new_class) ;
171                 q->next = NULL ;
172
173                 prev = q ;
174                 if (p->next == NULL)
175                     break ;
176                 else
177                     p = p->next ;
178             }
179             while (p->next != NULL)
180                 p = p->next ;
181
182             p->next = head ;
183             return OK ;
184         }
185         p = p->next ;
186     }
187     return ERROR ;
188 }
189
190
191 int rom_class(rom_class)
192 char *rom_class ;
193 {
194     int error = ERROR ;
195     SEG_DESCRIPTOR *p ;
196
197     /*
198      * Sets the romable field for the specified class to TRUE permitting
199      * the output of the segment in the absolute object file.
200      */
201
202     p = seg_list ;
203     while (p != NULL) {
204         if (strcmp(p->class, rom_class) == 0) {
205             p->romable = TRUE ;
206             error = OK ;
207         }
208         p = p->next ;
209     }
210     return error ;
211 }
212
213
214 int locate_virtual_segment(vseg, pseg)
215 unsigned int vseg ;
216 unsigned int *pseg ;
217 {
218     SEG_DESCRIPTOR *p ;
219
220     /*
221      * Finds an initialized segment with the specified virtual segment
222      * number and returns the corresponding physical segment number.
223      */
224
225     p = seg_list ;

```

(continued on next page)

GenView

3 D GRAPHICS ANIMATION

Now you can have a 3 D animating graphics system for your EGA or CGA equipped PC or PC compatible. (DOS 2.1 or higher.)

- 1 Includes all source code (C and ASM)
- 2 Hidden surface removal, translation, scaling and rotation
- 3 Scripting for viewpoint and object movements
- 4 Slideshow or animated playback of display frames
- 5 Software interface to single frame camera
- 6 "Fly through" of scenes with moving objects
- 7 Variable order spline curve generation for object and view point path generation
- 8 Object description includes MACRO capability
- 9 Source code includes direct to hardware assembly routines for high speed display
- 10 Control viewpoint path, acceleration, and velocity
- 11 Control object path, velocity, and rotation

GenView allows you to specify a viewpoint path with a small set of "turning points." It will then generate a smoothed series of frame locations along the path according to the velocity and acceleration parameters you specify. View direction may be backwards, forwards, or user specified. Object movement is controlled through a similar process with the addition of rotation (about any or all axes).

CGA Animation Sequence Demo
\$7.00

EGA Animation Sequence Demo
\$9.00 (2 diskettes)

CGA Flythrough Sequence Demo
\$9.00 (Requires hard disk)

GenView System \$99.00

The GenView System includes executable code, source, MAKE files, object description library, and manual (more than 40 pages) on diskette.

No Purchase Orders. Massachusetts residents add 5% sales tax.
Outside USA add \$15.00.
VISA and MASTERCARD accepted.

Call 617-528-4280 to order or send check or money order to:

GenSoft Systems

Dept. 9D

P.O. Box 1

Foxborough, MA 02035

CIRCLE 153 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Five (Listing continued, text begins on page 38.)

```
226 while (p != NULL) {
227     if (p->initd == TRUE && p->vseg == vseg && p->len != 0) {
228         *pseg = p->pseg ;
229         return OK ;
230     }
231     p = p->next ;
232 }
233 return ERROR ;
234 }
```

End Listing Five

Listing Six

OUTHEX.C

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <io.h>
4 #include <string.h>
5 #include <malloc.h>
6 #include <dos.h>
7
8 #include "loc.h"
9 #include "externs.h"
10
11
12 void output_hex_OMF(hex_file, seg_list, entry_point)
13 int hex_file ;
14 SEG_DESCRIPTOR *seg_list ;
15 unsigned char *entry_point ;
16 {
17     unsigned int offset, i, count ;
18     unsigned char *seg_start, *text ;
19
20     SEG_DESCRIPTOR *p ;
21
22     /*
23      * This function controls the sequencing of the Intel extended hex
24      * output using the Intel hex output routines.
25      */
26
27     /* Run through the segment list and output all ROMable segments */
28     p = seg_list ;
29     while (p != NULL) {
30         if (p->romable == TRUE) {
31
```

*C Programmers: Combine C and COMMON LISP
to Increase the Power of Your Software*

TransLISP PLUS™

Simple.

Add LISP features to your software without making it a full time job. The TransLISP PLUS tutorial, on-line help, and 30 sample programs with commented source make it easy.

Practical.

Start by modifying the LISP sample programs and including them in a system you wrote in C. Yes, in C! TransLISP PLUS includes a C Language Interface that lets you integrate your Microsoft C code and libraries with all or portions of our LISP interpreter.

Use TransLISP PLUS to add natural or command language features to replace menus ... or to flexibly manage related but disparate information. Code from C libraries provided by other vendors can be integrated into your program to perform tasks not normally part of LISP.

Thorough.

TransLISP PLUS took over 400 primitives from the most widely used and respected LISP standard, COMMON LISP, and made it available on IBM PCs, XTs, ATs, and virtually every other MSDOS machine. So now you can work with anything from a \$700 PC to a \$7000 PC.

The utilities toolbox is included at no charge with a built-in editor, pretty printer, cross reference, and additional debugging tools.

An optional Runtime encrypts your source code so that you can distribute your applications safely. You pay no royalties.

Requires MSDOS 2.0+, 320K RAM, and a 360K floppy.

MONEYBACK GUARANTEE

Try TransLISP PLUS (\$195) for 30 days — if not satisfied get a full product refund. The Optional Runtime is available for \$150. Or start by learning LISP with TransLISP (\$95) then upgrade to PLUS for \$158.

Call (800) 255-4659

In MA (617) 331-0800

CIRCLE 315 ON READER SERVICE CARD



**The
Coder's
Source™**

541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE 154 ON READER SERVICE CARD


```

32      /* Allocate enough memory to hold the segment (up to 64K) */
33      if ((text = get_mem((unsigned long) p->len)) == NULL) {
34          perror(_FILE_);
35          exit(1);
36      }
37
38      /* Locate the position of the segment in the load module file */
39      if (lseek(tmp_file, p->position, SEEK_SET) == -1L) {
40          perror(_FILE_);
41          exit(1);
42      }
43
44      /* Read in the segment and pad with zero if necessary */
45      count = read(tmp_file, text, p->len);
46      if (count != p->len) {
47          if (count == -1) {
48              perror(_FILE_);
49              exit(1);
50          }
51          else
52              memset(text + count, '\0', p->len - count);
53      }
54
55      /* Write the segment number out in an address record */
56      write_ADDR_record(hex_file, p->pseg);
57
58      /* Output the segment as a series of 16 byte data records */
59      offset = p->offset;
60      seg_start = text;
61      for (i = 0; i < p->len / 16; i++) {
62          write_DATA_record(hex_file, offset, seg_start, 16);
63          offset += 16;
64          seg_start += 16;
65      }
66
67      /* Handle any remaining data */
68      if ((p->len % 16) != 0)
69          write_DATA_record(hex_file, offset, seg_start, p->len % 16);
70
71      free_mem(text);
72  }
73  p = p->next;
74  }
75
76  /* Write the START and EOF records */
77  write_START_record(hex_file, entry_point);
78  write_EOF_record(hex_file);
79
80  return;

```

(continued on next page)

8031

FORTH DEVELOPMENT ENVIRONMENT

Take advantage of Bryte's tools to make your job easier:

- Bryte's development environment uses BRYTE-FORTH on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, detailed listings, and cross-references.

Why not start developing the Bryte way today?

BRYTE-FORTH 8031 EPROM	100.00
(includes 130 page User's Manual)	
Utility disk(s)	65.00*
Cross-compiler/Cross-assembler	235.00*
8031 unlimited quan. license	1000.00*

* Includes complete source code

bryte computers, inc.

P.O. Box 46
Augusta, ME 04330-0046

207/547-3218

CIRCLE 155 ON READER SERVICE CARD

Create 68K Embedded Systems On ATs

Oregon Software Brings VAX and VME
Pascal-2[®] Cross-Development Tools
To MS-DOS Workstations

Get All Your Tools In One Package

Compiler

- 68000, 68020, and 68881 instruction sets
- ROMable code

Assembler-Linker

Concurrent Program Support
Stand-Alone Library
Source Management

OREGON  SOFTWARE

800-367-2202

6915 SW Macadam Avenue, Portland, Oregon 97219 U.S.A.
503-245-2202 FAX 503-245-8449 TWX 910-464-4779

CIRCLE 156 ON READER SERVICE CARD

OUTSIDE OKLAHOMA NO SALES TAX

640K MOTHERBOARD UPGRADE: Zenith 150,
 IBM PC XT, Compaq Portable & Plus; hp Vectra

DYNAMIC RAM			
1Mbit	1048Kx1	100 ns	\$29.50
1Mbit	256Kx4	120 ns	32.00
51258	*256Kx1	100 ns	6.95
4464	64Kx4	150 ns	3.60
41256	256Kx1	80 ns	5.95
41256	256Kx1	100 ns	5.15
41256	256Kx1	120 ns	3.95
41256	256Kx1	150 ns	3.50
41264	+ 64Kx4	120 ns	5.25
EPROM			
27C1000	128Kx8	150 ns	\$37.95
27C512	64Kx8	200 ns	15.50
27256	32Kx8	250 ns	5.75
27128	16Kx8	250 ns	5.40
STATIC RAM			
43256L-12	32Kx8	120 ns	\$11.65
5565PL-15	8Kx8	150 ns	3.30

* 2-PORT
 VIDEO
 RAM
 * STATIC
 COLUMN
 DRAM
 FOR 386
 COMPAT
 80387-16
 80287-8
 \$245.00
 \$510.00
 \$160.00

OPEN 6 1/2 DAYS, 7-30 AM-10 PM. SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS
 RECEIVED BY: 24,000 S. Peoria Ave. (918) 267-4961
 BEGGS, OK. 74421
 No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to \$1 for packing materials. Orders received by 4 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ \$4.00, or guaranteed next day Priority One @ \$10.50!

CIRCLE 157 ON READER SERVICE CARD

Changing Your Address?

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

Name _____

Address _____

Apt. # _____ State _____

City _____

Zip _____

Mail to:

Dr. Dobb's Journal,
 PO Box 27809,
 San Diego, CA 92128

DOS LOCATE UTILITY

Listing Six (Listing continued, text begins on page 38.)

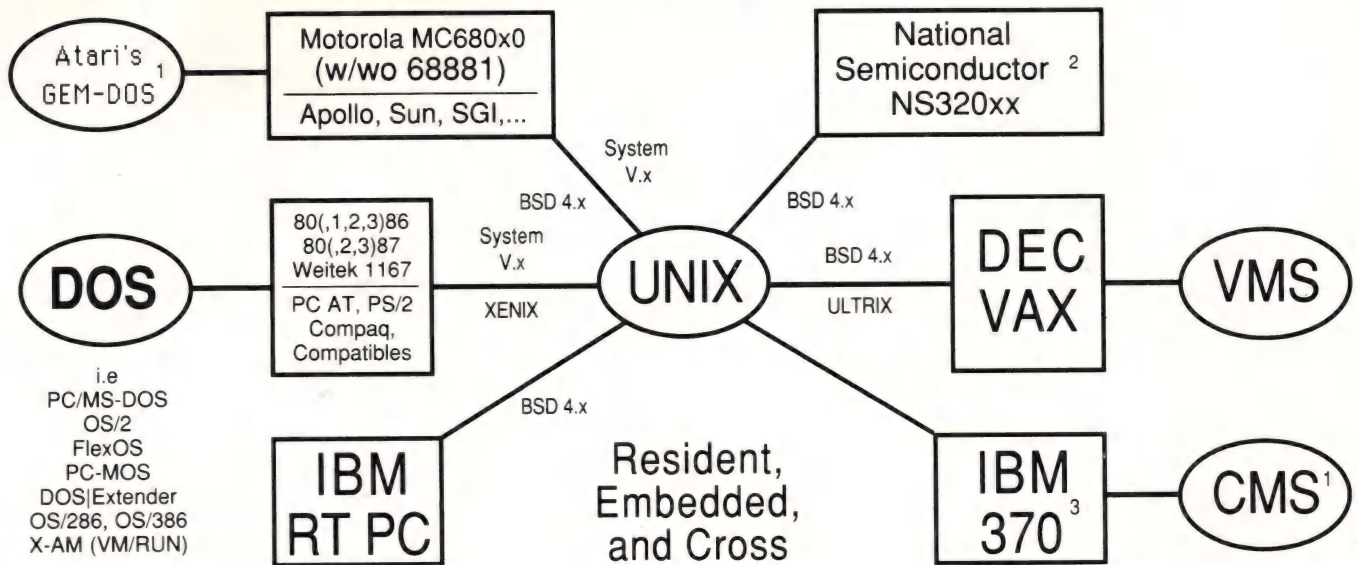
```

81 }
82
83
84 void write_ADDR_record(file, usba)
85 int file ;
86 unsigned int usba ;
87 {
88     unsigned char buf[16], *p ;
89     unsigned char len_field = 2 ;
90
91     /*
92      * This function writes an Intel extended hex Address record to the
93      * output file. Inputs are the file handle and the USBA (segment
94      * base address).
95      */
96
97     p = buf ;
98     *p++ = high_byte(usba) ;
99     *p++ = low_byte(usba) ;
100
101     output_hex_record(file, ADDR_RECORD, 0, buf, p - buf) ;
102
103     return ;
104 }
105
106 void write_EOF_record(file)
107 int file ;
108 {
109     /*
110      * This function writes an Intel extended hex EOF record to the
111      * output file.
112      */
113
114     output_hex_record(file, EOF_RECORD, 0, NULL, 0) ;
115     return ;
116 }
117
118
119 void write_DATA_record(file, offset, text, len)
120 int file ;
121 unsigned int offset ;
122 unsigned char *text ;
123 unsigned int len ;
124 {
125
126     /*
127      * This function writes an Intel extended hex Data record to the
128      * specified output file.
129      */
130
131     output_hex_record(file, DATA_RECORD, offset, text, len) ;
132
133     return ;
134 }
135
136 void write_START_record(file, entry)
137 int file ;
138 unsigned char *entry ;
139 {
140     unsigned char *buf, *p ;
141     unsigned int count ;
142
143     unsigned char len_field = 4 ;
144     unsigned int addr_field = 0 ;
145     unsigned char rec_type = START_RECORD ;
146
147     /*
148      * This function writes an Intel extended hex Start record to the
149      * output file.
150      */
151
152     /* Allocate some memory to build the data field in */
153     if ((p = buf = (unsigned char *) malloc(32)) == NULL) {
154         perror(_FILE_) ;
155         exit(1) ;
156     }
157
158     /* Store the start address in the data field (segment first) */
159     *p++ = high_byte(FP_SEG(entry)) ;
160     *p++ = low_byte(FP_SEG(entry)) ;
161
162     /* And then the offset */
163     *p++ = high_byte(FP_OFF(entry)) ;
164     *p++ = low_byte(FP_OFF(entry)) ;
165
166     /* Output the record */
167     output_hex_record(file, START_RECORD, 0, buf, p - buf) ;
168
169     free(buf) ;
170     return ;
171 }
172
173
174 void output_hex_record(file, type, addr, data, length)
175 int file ;
176 unsigned char type ;
177 unsigned int addr ;
178 unsigned char *data ;
179 unsigned char length ;
180 {
181     char *p, *buf ;
182     unsigned int size, count ;
183     unsigned char chksum, digit ;

```

(continued on page 86)

C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper™** utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers:

Professional Pascal™ and **High C™** on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal*; *PC Tech Journal*; *PC Magazine*; and the *Journal of Pascal, Ada, and Modula-2*.

We supplied the **first** compilers generating **32-bit protected-mode code for the 80386** under MS-DOS (since 11/86). And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**HC v1.4 and PP v2.7 released May 1987.**)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal's** Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS, ...

Professional developers in need of industrial-strength tools contact:



MetaWare Incorporated
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060-4429

(408) 429-6382

Telex: 493-0879 (META UI)
PC Tech Journal's conclusion:

The Clear Choice for Large Programming Projects.

Name _____
Company _____
Address _____
City, ST _____ Zip _____
Phone (____) _____

Product:
Platform:

Circle what interests you:

PP HC TWS Helper (DOS only)
V.x 4.x DOS FlexOS VMS CMS
Sun Apollo Atari VAX 370
8086-family 80386 680x0 32032

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others/ owners: Ada/DoD; Apollo/Apollo; Atari/Atari; DEC/VAX,VMS/DEC; FlexOS,GEM-DOS/DRI; IBM,RT PC/IBM; MS-DOS/Microsoft; Q&A/Symantec; Sun/Sun Microsystems; UNIX/AT&T.

Footnotes: 1. Atari, CMS versions available 10/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

CIRCLE 158 ON READER SERVICE CARD

Listing Six (Listing continued, text begins on page 38.)

```

186
187 /*
188 This function does all of the work of writing an Intel extended
189 hex output record. The inputs to this routine are:
190 file      output file handle
191 type      record type
192 addr      address field value
193 data      data field contents
194 length    size of the data field
195 */
196
197 /* Allocate some memory to build the output record in */
198 if ((p = buf = malloc(550)) == NULL) {
199     perror(_FILE_);
200     exit(1);
201 }
202
203 /* Build the prefix for the data field */
204 p += sprintf(p, ":%02X%02X%02X%02X", length, high_byte(addr), \
205             low_byte(addr), type);
206
207 /* Compute the checksum on the prefix */
208 chksum = length + high_byte(addr) + low_byte(addr) + type;
209
210 /* Build the data field byte by byte */
211 while (length--) {
212     digit = (*data >> 4) & 0x0f;
213     *p++ = (digit > 9) ? digit + 0x37 : digit + '0';
214     digit = *data & 0x0f;
215     *p++ = (digit > 9) ? digit + 0x37 : digit + '0';
216     chksum += *data++;
217 }
218
219 /* Compute the complement of the checksum and output */
220 chksum = ~chksum + 1;
221 p += sprintf(p, "%02X\r\n", chksum);
222
223 /* Compute the size of the output record and output */
224 size = p - buf;
225 count = write(file, buf, size);
226 if (count != size) {
227     perror(_FILE_);
228     exit(1);
229 }
230
231 free(buf);
232 return;
233 }

```

End Listing Six**Listing Seven**

PRINTLOC.C

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <time.h>
5
6  #include "loc.h"
7  #include "externs.h"
8
9
10 int  print_statistics(map_filename, stat_filename, command_line, \
11                      exename, output_file, configname, entry_point)
12 char *map_filename;
13 char *stat_filename;
14 char *command_line;
15 char *exename;
16 char *output_file;
17 char *configname;
18 unsigned char *entry_point;
19 {
20     unsigned long temp;
21     long ltime;
22     int i;
23     SEG_DESCRIPTOR *p;
24     SYMBOL_LIST *q;
25
26     /*
27      This function generates the locate map file. The locate map
28      file contains the segment information with the physical
29      segment addresses.
30     */
31
32     fprintf(print_file, "MS-DOS Locate Utility Version 1.0\n\n");
33     fprintf(print_file, "Input File: %s\n", exename);
34     fprintf(print_file, "Output File: %s\n", output_file);
35     fprintf(print_file, "Configuration File: %s\n", configname);
36     fprintf(print_file, "Invoked by: %s\n", command_line);
37
38     time(&ltime);
39     fprintf(print_file, "Date/Time: %s\n", ctime(&ltime));
40
41     /* Display the located segment information */
42     fprintf(print_file, "Segment information\n");
43     fprintf(print_file, "%-16s%-16s%-12s%-12s\n", "Name", "Class",
44             "Address", "Length");
45
46     p = seg_list;
47     while (p != NULL) {
48         temp = (unsigned long) p->pseg;
49         temp = temp * 16 + p->offset;
50         fprintf(print_file, "%-16s%-16s%05lX%10.04lX\n", p->name,
51                 p->class, temp, p->len);
52         p = p->next;
53     }

```



```

54
55 /* Display the symbol information */
56 fprintf(print_file, "\n\nPublic Symbols\n");
57 i = 0 ;
58 p = seg_list ;
59 while (p != NULL) {
60     q = p->symbol_list ;
61     while (q != NULL) {
62         fprintf(print_file, "%04X:%04X %16s%s", p->pseg,
63             q->value, q->name, i++ ? "\n" : "\t\t");
64         i %= 2 ;
65         q = q->next ;
66     }
67     p = p->next ;
68 }
69
70 fprintf(print_file, "%sEntry Point - %p\n", (i == 1) ? "\n\n" : "\n",
71     entry_point) ;
72
73 return ;
74 }

```

End Listing Seven

Listing Eight

READCFG.C

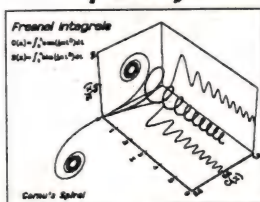
```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "loc.h"
6  #include "externs.h"
7
8  int  process_class_keyword(), process_order_keyword(), process_null();
9  int  process_rom_keyword(), process_dup_keyword(), process_comment();
10
11
12 static struct CFG_COMMANDS {
13     char *cmd ;
14     int (*command)();
15 } cfg_cmds[] = {
16     "CLASS",    process_class_keyword,
17     "ORDER",    process_order_keyword,
18     "ROM",      process_rom_keyword,
19     "DUP",      process_dup_keyword,
20     ":",        process_comment
21 };
22
23
24 int  process_locate_file(seg_list)

```

(continued on next page)

GraphiCTM
can plot your
data in
publication
quality



on your IBM PC

- linear, log, polar plots
- bar charts, Smith charts
- 3D curves, 3D surfaces
- 6 curve types, 8 markers
- 14 fonts, font editor
- 4096 x 3120 resolution
- zoom, pan, window plots
- high resolution printer and plotter dumps in color

Over 150 C and Assembler routines for full control

\$395 with source code

For personal use only

MOST HARDWARE IS SUPPORTED

Scientific Endeavors Corporation

Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

VTEKTM

DECTM VT100/VT52
and TektronixTM
4010/4014/4105
Terminal Emulator

- 20 user-defined keys
- scroll back buffer
- hardware 132 columns
- Kermit and XMODEM
- up to 800x600 screen resolution on EGAs
- zoom, pan, window plots
- "hot key" to DOS
- enhanced keyboard support
- ANSI extensions to VT100 for multi-color text
- scrolling VT100 window on graphics screen

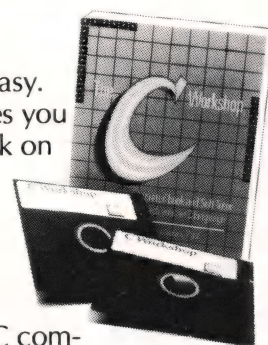
\$150. Site and source code licenses available

B-EDITTM

Our new binary editor for programmers - \$29

Easy to C

C is a great programming language. Now the new **C WORKSHOP** makes it easy. Interactive software teaches you C with immediate feedback on your program exercises. The **C WORKSHOP** has everything you need to learn C and write your own programs, too. You get a fast editor, standard C compiler, and online help. Plus our renowned book. Let the other guy struggle with confusing books and compilers. Join AT&T and other major companies that use the **C WORKSHOP**. InfoWorld columnist Adam Green calls it "the most intriguing new type of training system I've ever seen."



Quality software since 1981

Satisfaction Guaranteed

or return in 30 days for refund. To order the **C WORKSHOP**, call toll-free (800) 227-2400 ext. 955 day or night (Visa, MC, or AmEx). Or send check to Wordcraft, 3827 Penniman Av, Oakland, CA 94619. \$69.95 plus \$5.00 shipping (Priority Mail). In CA, add \$4.90 sales tax.

CIRCLE 160 ON READER SERVICE CARD

CIRCLE 159 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Eight (Listing continued, text begins on page 38.)

```
25 SEG_DESCRIPTOR *seg_list ;
26 {
27     int i, error = OK;
28     char *tok, *buf ;
29
30     /*
31      * This function reads the configuration file and performs the parsing
32      * and control transfer to routines which perform the desired action.
33      */
34
35     /* Allocate some memory for the line buffer */
36     if ((buf = malloc(256)) == NULL) {
37         perror(__FILE__) ;
38         exit(1) ;
39     }
40
41     /* Read and categorize a token from the configuration file */
42     while (fgets(buf, 256, config_file) != NULL) {
43         /* Extract the first token (read the next line if none is found */
44         if ((tok = strtok(buf, " \t\n")) == NULL)
45             continue ;
46
47         for (i = 0; i < dim(cfg_cmds); i++) {
48             if (strcmp(cfg_cmds[i].cmd, tok) == 0) {
49                 error = (cfg_cmds[i].command) () ;
50                 break ;
51             }
52         }
53
54         if (i == dim(cfg_cmds)) {
55             fprintf(stderr, "Illegal input - < %s>\n", tok) ;
56             exit(1) ;
57         }
58     }
59     free(buf) ;
60     return error ;
61 }
62
63 int process_comment()
64 {
65     return OK ;
66 }
67
68
69 int process_class_keyword()
70 {
71     char *tok, name[32], *p ;
72     unsigned int seg ;
73
74     /*
75      * This function parses the remainder of the CLASS directive.
76      */
77
78     /* Read the class name */
79     strcpy(name, strdup(strtok(NULL, " \t\n="))) ;
80
81     /* Verify that an equal sign is present */
82     if (strcmp((tok = strtok(NULL, " \t\n")), "=") != 0) {
83         fprintf(stderr, "\"=\" expected instead found < %s>\n", tok) ;
84         return ERROR ;
85     }
86
87     /* Read the segment number for the class */
88     tok = strtok(NULL, " \t\n") ;
89     seg = (unsigned int) strtoul(tok, &p, 0) ;
90     if (*p) {
91         fprintf(stderr, "Unrecognized token < %s>\n", p) ;
92         return ERROR ;
93     }
94
95     /* Assign the physical segment number to the specified class */
96     if (assign_physical_segment(name, seg) == ERROR) {
97         fprintf(stderr, "Undefined class < %s>\n", name) ;
98         return ERROR ;
99     }
100
101     return OK ;
102 }
103
104
105 int process_order_keyword()
106 {
107     char *tok, pclass[32], class[32] ;
108     unsigned int next_seg ;
109     BOOLEAN found = FALSE ;
110
111     /*
112      * This function processes the ORDER directive.
113      */
114
115     /* Read the leading class name from the command */
116     strcpy(pclass, strdup((tok = strtok(NULL, " \t\n")))) ;
117
118     /* Process the remaining class names in the command */
119     while ((tok = strtok(NULL, " \t\n")) != NULL) {
120         if (*tok == ';')
121             break ;
122
123         found = TRUE ;
124         strcpy(class, strdup(tok)) ;
125
126         /* Compute the segment address for this class to be made
127          * contiguous with the previous class */
128         if (get_next_segment(pclass, class, &next_seg) == ERROR) {
129
130
```

(continued on page 90)

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.
27511-8000. Telephone (919) 467-8000, x 7000

DDJ12/87

function libraries
disassemblers
compilers
text editors
text filters
communications support
text formatters
interpreters
bulletin boards
co-routines
compiler compilers
window packages
assemblers
games
tutorials
math packages
link editors
languages
cross compilers
pre-processors
function libraries
disassemblers
compilers
text editors

The C Users' Group Library

A Directory
of Public Domain
C Source Code

Send \$10
for Directory. Write
or call for more details
on over 100 volumes of
Public Domain C Source
Code.

The C Users' Group
PO Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE 162 ON READER SERVICE CARD

NEW! TLIB™ 4.0 SOURCE CODE CONTROL

The best keeps getting better!

The critics loved TLIB 3.0...

"...packed with features... [generates deltas] amazingly fast... [of the 6 reviewed] the two best packages are Burton Systems' TLIB and [a \$395 product], so designated because of their ease of use, abundance of features, and ability to be configured..." **PC Tech Journal Sept 87**

"...has my highest recommendation." **Ronny Richardson, Computer Shopper Aug 87**

- The fastest, most powerful system is now even faster!
- **Many new features!** Expanded keyword support. Multi-line comments. Branching, for multiple development lines. Extended wildcard and list-of-file support; creates lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support.
- Includes a copy of Landon Dyer's excellent public domain **MAKE** utility (with source code for DOS & VAX/VMS).

PC/MS-DOS 2.x & 3.x **Just \$99.95 + \$3 s/h** Visa/MC

BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27519-4156
(919) 469-3068

CIRCLE 163 ON READER SERVICE CARD

Announcing ! ! !

XenoFont

text screen printing

utilities for laser

printing

highest quality

accurate images

boldface reverse

cursor on/off

great for documentation

for only \$49.95 +S/H

XenoCopy PC

PC-DOS program lets your PC
Read/Write/Format over 300 formats

\$79.95 + \$5.00 S/H Sales Tax if CA.

XENOSOFT™

1454 Sixth Street, Berkeley, CA 94710



(415) 525-3113



CIRCLE 164 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Eight (Listing continued, text begins on page 38.)

```

131     fprintf(stderr, "Undefined class <%s>\n", pclass);
132     return ERROR;
133 }
134
135 /* Assign the computed segment number to the class */
136 if (assign_physical_segment(class, next_seg) == ERROR) {
137     fprintf(stderr, "Undefined class <%s>\n", class);
138     return ERROR;
139 }
140
141 /* Setup to process the next class */
142 strcpy(pclass, class);
143 }
144
145 return (found == FALSE) ? ERROR : OK;
146 }
147
148 int process_dup_keyword()
149 {
150     char *tok, old_class[32], new_class[32];
151
152     /* This function is responsible for processing the DUP directive.
153     */
154
155     /* Read the existing class name */
156     strcpy(old_class, strtok(NULL, " \t\n"));
157
158     /* Read the name of the class to be created */
159     strcpy(new_class, strtok(NULL, " \t\n"));
160
161     /* Duplicate the existing class */
162     if (dup_class(old_class, new_class) == ERROR) {
163         fprintf(stderr, "Undefined class <%s>\n", old_class);
164         return ERROR;
165     }
166     return OK;
167 }
168
169 int process_rom_keyword()
170 {
171     char *tok, class[32];
172
173     /* This function processes the ROM keyword and marks all specified
174     classes as ROMable.
175     */
176
177     /* Read all of the tokens on the line */
178     while ((tok = strtok(NULL, " \t\n")) != NULL) {
179         if (*tok == ';')
180             break;
181         strcpy(class, strtok(NULL, " \t\n"));
182         if (rom_class(class) == ERROR) {
183             fprintf(stderr, "Undefined class <%s>\n", class);
184             return ERROR;
185         }
186     }
187     return OK;
188 }
189
190
191
192
193
194
195
196
197

```

End Listing Eight

Listing Nine

READMAP.C

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <malloc.h>
5
6  #include "loc.h"
7  #include "externs.h"
8
9  #define BUFSIZE 256
10
11
12 SEG_DESCRIPTOR *build_seg_list()
13 {
14     int count;
15     unsigned long start_seg, end_seg, length;
16     char seg_name[32], class[32];
17     char *buf;
18
19     SEG_DESCRIPTOR *p, *previous, *list_start, *class_start;
20
21     /*
22     This function is responsible for the processing of the link map.
23     The link map is read and the segment information such as segment
24     name, segment length and class name are recorded.
25     */
26
27     /* Seek to the beginning of the file */
28     if (fseek(map_file, 0L, SEEK_SET) != 0) {
29         perror("_FILE_");
30         exit(1);
31     }

```



```

32
33 /* Allocate some memory for the line buffer */
34 if ((buf = malloc(BUFSIZE)) == NULL) {
35     perror(_FILE_);
36     exit(1);
37 }
38
39 /* Search thru the file looking for the start of the segment informati/
40 while (1) {
41     if (fgets(buf, BUFSIZE, map_file) == NULL) {
42         fprintf(stderr, "Unable to find the segment list in %s\n", map_f;
43         exit(1);
44     }
45
46     if (strstr(strupr(buf), "START") != NULL)
47         break;
48 }
49
50 /* Scan to the start of the first segment record */
51 while (fgets(buf, BUFSIZE, map_file) != NULL) {
52     count = sscanf(buf, "%lxH %lxH %lxH %s %s", &start_seg, &end_seg, ;
53     if (count == 5)
54         break;
55 }
56
57 /* Check if EOF was detected and an error message should be printed */
58 if (feof(map_file)) {
59     fprintf(stderr, "Unable to find the segment list in %s\n", map_fname;
60     exit(1);
61 }
62
63 /* Begin processing the list of segments */
64 p = previous = NULL;
65 while (count -- 5) {
66     /* Allocate some memory to hold the data structure */
67     if ((p = (SEG_DESCRIPTOR *) malloc(sizeof (*p))) == NULL) {
68         perror(_FILE_);
69         exit(1);
70     }
71
72     if (previous == NULL)
73         list_start = p;
74     else
75         previous->next = p;
76
77     strcpy(p->name, strupr(seg_name));
78     strcpy(p->class, strupr(class));
79     p->vseg = (unsigned int) start_seg / 16;
80     p->offset = (unsigned int) start_seg % 16;
81     p->len = (unsigned int) length;
82     p->position = start_seg;
83     p->inited = FALSE;
84     p->romable = FALSE;
85     p->symbols = 0;
86     p->symbol_list = NULL;
87     p->next = NULL;
88
89     /* Check if the class name has changed and reset the offset */
90     if (strcmp(p->class, class_start->class) != 0) {
91         p->pseg = 0;
92         class_start = p;
93     }
94     else
95         p->pseg = p->vseg - class_start->vseg;
96
97     previous = p;
98
99     /* Read the next line of segment information */
100    fgets(buf, BUFSIZE, map_file);
101    count = sscanf(buf, "%lxH %lxH %lxH %s %s", &start_seg, &end_seg, ;
102 }
103
104 free(buf);
105 return (list_start);
106 }

```

End Listing Nine

Listing Ten

```

SIEVE.C
1 /*
2 Sieve Benchmark - ROM Version
3
4 Copyright (C) Recycled Software 1987. All rights reserved.
5
6 Executes 100 iterations of the sieve algorithm for microprocessor
7 benchmarking purposes.
8
9 */
10
11
12 #define TRUE 1
13 #define FALSE 0
14 #define SIZE 8190
15
16 char flags[SIZE + 1];
17
18 main()
19 {
20     int i, prime, k, count, iter;
21

```

(continued on next page)

A Reconfigurable Programmer's Editor With Source Code

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- Multiple Windows
- Column cut and paste
- Line marking for source code
- Regular Expressions
- C-like Macro Language
- Reconfigurable Keyboard
- Capture your DOS session
- Run your compiler and examine errors
- Comes with useful precompiled macros
- UNIX-like CTAGS
- 43-line mode with EGA

This is the **ONLY** editor with the power of the higher priced editors which comes with complete source code!

New commands and features may be added to the editor by writing programs in its macro language. The language resembles C, and comes with a rich set of primitives for handling strings and changing the editor environment, plus most of the flow-of-control constructs that come in C (*for, while, if, break, continue*).

The code is written in standard C, with several key library routines written in assembler for speed. The source code option is perfect for OEMs and VARs who want to add editing or word processing capabilities to their applications.

Price for editor and on-line documentation ---- \$39.95

Price for editor with complete source code -- \$94.95

(Please add \$2 for shipping & handling, \$6 overseas)

Special offer -- New York Word word processor -- \$39.95 -- Multi-windowing, mail merge, hyphenation, math, regular expressions, TOC and index generators

MAGMA SYSTEMS

138-23 Hoover Ave. Jamaica, NY
11435 (201) 792-3954

CIRCLE 165 ON READER SERVICE CARD

DOS LOCATE UTILITY

Listing Ten (Listing continued, text begins on page 38.)

```

22  for (iter = 1; iter <= 100; iter++) {
23      count = 0 ;
24      for (i = 0; i <= SIZE; i++)
25          flags[i] = TRUE ;
26
27      for (i = 0; i <= SIZE; i++) {
28          if (flags[i]) {
29              prime = i + 1 + 3 ;
30              for (k = i + prime; k <= SIZE; k += prime)
31                  flags[k] = FALSE ;
32
33              count++ ;
34          }
35      }
36  }
37  }
38
39

```

End Listing Ten

Listing Eleven

STABLE.C

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <dos.h>
5
6  #include "loc.h"
7  #include "externs.h"
8
9  char *errstr = "Unable to locate global symbol \"%s\\\"\\n\" ;
10
11
12 void read_symbol_table(seg_list)
13 SEG_DESCRIPTOR *seg_list ;
14 {
15     char buf[128] ;
16     int count, found ;
17     unsigned int vseg, off;
18     char symbol[32], attrb[10] ;
19
20     SEG_DESCRIPTOR *p ;
21     SYMBOL_LIST *q ;
22
23     /*
24      * This function reads the linker map file and extracts the
25      * symbol information.

```

(continued on page 94)

C_talk™ OBJECT-ORIENTED PROGRAMMING IN C

What is C_talk™

C_talk™ is an object-oriented development environment in C with Smalltalk-like messaging formats. It lets the software developer use the C_talk™ Browser to develop an object-oriented program, then use the C_talk™ Compiler to convert this program into C code compatible with most popular C compilers. The combined data abstraction power of object-oriented programming with the efficiency, speed, and flexibility of C results in a high productivity development and delivery environment which can significantly cut development time. C_talk™ offers:

The Power of OOLs

C_talk™ is designed to let you take full advantage of object-oriented languages (OOLs). It is designed so that both the object-oriented guru and the "non-objecting" neophyte can use C_talk™ to explore and exploit the exciting world of OOLs. C_talk™ contains:

- Encapsulation ("objects")
- Messaging
- Inheritance

The Efficiency of C

C_talk™ does just what its name suggests - lets you talk in C, and thereby gives you all the efficiency and advantages of C:

- Speed, Size, Flexibility
- Ease of Application Delivery
- Access to C libraries and C tool sets

The user of standard C will find programming in C_talk™ is basically programming in C, but with a powerful difference: C_talk™ is an object-oriented environment. C_talk™ introduces to C a new data type - the object, and a new operation - the message.

The Productivity of C_talk™

C_talk™ is a synergy of C and Smalltalk-like features - yielding much greater productivity to the software builder:

- Define software components in object-oriented terms.
- Extend software components with full class-inheritance.
- Automatically convert work into C code.
- Reuse software components to obtain results in less time.
- Learn quickly using standard C in C_talk™.

System Requirements

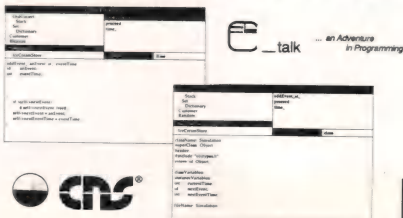
C_talk™ (version 1.0) is designed to run on the IBM® PC (or compatibles) with graphics (CGA, EGA, VGA) and with one of the following C languages: Microsoft® C, Lattice C, Turbo C, or C86. A system configured with a hard drive and mouse is highly recommended.

TO ORDER:

CNS, Inc.
Software Products Dept.
7090 Shady Oak Road
Eden Prairie, MN 55344
(612) 944-0170

PRICE: \$149.95
Credit Cards: Master Card, Visa

IBM is a registered trade mark of IBM Corp.
MICROSOFT is a registered trade mark
of MICROSOFT CORP.
C_talk is a trade mark of CNS, Inc.



CIRCLE 167 ON READER SERVICE CARD

PRODUCTION LANGUAGES CORP.

PRODUCTION QUALITY 68020 ANSI C Compiler

- Full 68020 instruction set
- Full 68881 support
- ANSI Standard reentrant library
- Extensive Sybolic Debug information
- Internal consistency checking
- Position independent code
- ROMable code

PC Hosted cross-compiler

2 user license \$1500.00

VME Hosted native compiler

single CPU license..... \$1500.00

ANSI standard reentrant

library source code \$800.00

SPECIAL OFFER

For a limited time only we are offering full
library source FREE with the purchase of either
68020 C compiler

817-599-8366

P.O. Box 109, Weatherford, Texas 76086

CIRCLE 166 ON READER SERVICE CARD

It's good
for your system!

Vitamin C

"If you need source code, make sure
your wallet is wide open or get
VITAMIN C."

Picking the best value package is hard...
If you're a source code fanatic like me,
VITAMIN C is preferable."

- Computer Language, June, 1987

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination software professionals demand to produce superior applications in dramatically less time. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Extended versions of these routines add flexible control over specific details when necessary. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C \$225
Includes all source code FREE! For IBM
PC, XT, AT, PS/2 and true compatibles.
Specify compiler when ordering.

VCScreen \$99.95
For IBM PC, XT, AT, PS/2 and true
compatibles. Requires Vitamin C.

We ship UPS. Please include \$3 for
ground, \$6 for 2-day air, \$20 for
overnight, or \$30 if outside the U.S.
Texas residents add 7 1/4 % sales tax. All
funds MUST be in U.S. dollars drawn on a
U.S. bank. Visa & MasterCard accepted.

ORDER NOW!
(214)416-6447

creative
PROGRAMMING

Box 112097 • Carrollton, Texas 75011

- ✓ Professional C function library
- ✓ 30 day money back guarantee
- ✓ Multiple bullet proof windows
- ✓ Easy full screen data entry
- ✓ Unlimited data validation
- ✓ Context sensitive help manager
- ✓ Menus like Lotus and Mac
- ✓ Programmable keyboard handler
- ✓ Text editor routines
- ✓ No royalties or runtime fees
- ✓ Library source included FREE
- ✓ Free technical support
- ✓ Free BBS at (214)418-0059
- ✓ Supports all major compilers
including Microsoft 5.0
- ✓ VCScreen code generator too!
- ✓ UNIX version available,
call for details

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

DOS LOCATE UTILITY

Listing Eleven (Listing continued, text begins on page 38.)

```

26  */
27
28  /* Seek to the beginning of the file */
29  if (fseek(map_file, 0L, SEEK_SET) != 0) {
30      perror("FILE");
31      exit(1);
32  }
33
34  /* Search thru the file for the symbol tables */
35  while (1) {
36      if (fgets(buf, sizeof(buf), map_file) == NULL)
37          return;
38
39      if (strstr(strupr(buf), "ADDRESS") != NULL)
40          break;
41  }
42
43  /* Read each of the symbol entries */
44  while (1) {
45      count = fscanf(map_file, " %4x:%4x%5c %s", &vseg, &off, attrb, symb;
46      if (count != 4)
47          break;
48      else {
49          p = seg_list;
50          found = FALSE;
51          while (p != NULL) {
52              if ((p->vseg != vseg) || (p->len == 0)) {
53                  p = p->next;
54                  continue;
55              }
56
57              q = (SYMBOL_LIST *) malloc(sizeof(*q));
58              strcpy(q->name, symbol);
59              q->value = off;
60              q->type = 1;
61              q->next = p->symbol_list;
62              p->symbol_list = q;
63              p->symbols++;
64              found = TRUE;
65              break;
66          }
67
68          if (found == FALSE)
69              fprintf(stderr, errstr, symbol);
70      }
71  }
72
73  return;
74 }
75

```

End Listing Eleven

(Listings will continue next month)

POWER UP YOUR 386

MULTI-USER DOS UNDER UNIX®

In Stock - Ready to Ship

386/ix Applications Platform

386/ix UNIX V.3 run time

VP/ix (DOS under UNIX)

TEN/PLUS User Interface

Software Development System

INTRODUCTORY PRICE FOR TWO-USER SYSTEMS

All of the Above for \$894

Unlimited user licenses and more options available

Call now to order:

(800) 323-8649 or (312) 987-4084

COMPUTER TECHNOLOGY GROUP

Telemedia, Inc.

310 S. Michigan Ave., Chicago, IL 60604

UNIX is a registered trademark of AT&T
MS-DOS is a trademark of MICROSOFT
386/ix, VP/ix, and TEN/PLUS are trademarks of
INTERACTIVE SYSTEMS CORPORATION

CIRCLE 169 ON READER SERVICE CARD



Break The dBase Barrier

*dBase to C conversion
is now a reality*

Sooner or later you're going to run into the dBase wall. It may come up unexpectedly. Maybe you know it's there. But at some point you are going to need faster run-time, real portability, stronger code refinement, and source code security.

Using dBx to translate your dBase code to C is the perfect way to break the dBase barrier. C is portable, fast, and flexible. C programmers appreciate our commented, clean K&R code. If you are new to C, dBx is a great way to get up to speed. Why not call us today and discuss your individual situation.

dBx - The dBase to C Translator - It's Real



Desktop Ai

1720 Post Road E
Westport, CT 06880

(203) 255-3400

TELEX • 6502972226MCI



CIRCLE 170 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

FREE Innovative Software Technology Details

Since 1983, we've kept microcomputer developers abreast of software development trends. Our specialists help you with information about products that raise your productivity and enrich your programming environment. Now you can receive a special packet covering one of the 7 important approaches to productivity enhancement, PLUS a Free series of articles from our newsletter, "The Programmer's Letter," discussing this important subject. Call TODAY and specify Translators, Cross Compilers, 386 Native Mode Development, Prototyping Software, Object-Oriented Programming, Visual Programming, or Windowing Environments.

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7PM to 7AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

AI-Expert System Dev't

Arity Combination Package	PC	\$ 979
System - use with C	MS	\$ 229
CxPERT - shell for C	MS	\$ 259
Exsys	PC	\$ 289
Level 5 - formerly Insight 2 +	MS	\$ 589
T.I.: PC Easy	PC	\$ 435
Personal Consultant Plus	PC	\$2589
Turbo Expert-Startup (400 rules)	PC	\$ 129
Corporate (4000 rules)	PC	\$ 359

AI Languages

APT - Active Prolog Tutor - build applications interactively	PC	\$ 49
ARITY Prolog - full, 4 Meg		
Interpreter - debug, C, ASM	PC	\$ 229
COMPILER/Interpreter-EXE	PC	\$ 569
MicroProlog Prof. Comp./Interp.	MS	\$ 439
PC Scheme LISP - by TI	PC	\$ 85
Star Sapphire	MS	\$ 459
TransLISP - learn fast	MS	\$ 79
TransLISP PLUS	MS	\$ 149
TURBO PROLOG by Borland	PC	\$ 69
Others: IQ LISP (\$239), IQC LISP (\$269)		

Basic

BAS_C - economy	MS	\$ 179
BAS_PAS - economy	MS	\$ 135
Basic Development Tools	PC	\$ 89
Basic Windows by Syscom	PC	\$ 95
BetterBASIC	PC	\$ 129
Exim Toolkit - full	PC	\$ 45
Finally - by Komputerwerks	PC	\$ 85
Inside Track	PC	\$ 49
Mach 2 by MicroHelp	PC	\$ 55
QBase	MS	\$ 79
QuickBASIC	PC	\$ 69
Quick Pak-by Crescent Software	PC	\$ 59
Quick-Tools by BC Associates	PC	\$ 109
Stay-Res	PC	\$ 59
True Basic	PC	\$ 79
Turbo BASIC - by Borland	PC	\$ 69
Turbo BASIC Database Toolbox	MS	\$ 69

FEATURES

The Baler - compiles Lotus 1-2-3 (1A, 2.01, or VP-Planner). Protect code speed execution, save disk space, password capability, user does not need 1-2-3. **No Royalties.** PC \$ 459

MKS AWK by Mortice Kern Systems. "4GL" for data transformation and report generation (conforms to Bell Labs spec for UNIX System V.3) allows multiple-subscripted arrays. MS \$ 65

SPECIAL PRICES 386 TOOLS



DEVELOP 7 TIMES FASTER!

HUMMINGBOARD 386 - Develop 2.6 or 7.9 times faster than a 8 MHZ AT. AT or XT addin board uses dual processors for Speed and Hardware Debugging. 16 MHZ or 20 MHZ. Call about Benchmarks, Trial Program.

Call before December 31, 1987 and mention this ad for these SPECIAL PRICES:

	List	Normal	SPECIAL
DesqView PS/2	\$ 130	\$ 109	\$ 89
PC/MOS/386	\$ 195	\$ 179	\$ 159
XENIX System V			
Complete System	\$1495	\$1149	\$1095
Development System	\$ 695	\$ 589	\$ 559
Operating System	\$ 695	\$ 589	\$ 559

RECENT DISCOVERY

Windows/386 by Microsoft - multitask standard DOS applications in separate 640K segments and access expanded memory. Toggle, run simultaneously, or foreground only. PC \$149

C Language-Compilers

AZTEC C86 - Commercial	PC	\$499
C86 PLUS - by CI	MS	\$359
Datalight C - fast compile	PC	\$ 77
Datalight Optimum - C	MS	\$ 99
Lattice C - from Lattice	MS	\$269
Microsoft C 5.0- Codeview	MS	\$275
Microsoft Quick C	MS	\$ 69
Rex - C/86 standalone ROM	MS	\$695
Turbo C by Borland	PC	\$ 69

C Libraries-Files

BTree by Soft Focus	MS	\$ 69
CBTREE - Source, no royalties	MS	\$ 99
ctree by Faircom - no royalties	MS	\$315
rtree - report generation	PC	\$239
dB2C Toolkit V2.0	MS	\$249
dbQUERY - ad hoc, SQL-based	MS	Call
dbVISTA - Object only	MS	Call
Source - Single user	MS	Call
dBx - translator	MS	\$299

C-Screens, Windows, Graphics

C Worthy Interface Library	PC	\$249
Curses by Aspen Scientific	PC	\$109
dBASE Graphics for C	PC	\$ 69
ESSENTIAL GRAPHICS - fast	PC	\$185
FontWINDOW/PLUS	PC	\$229
GraphiC - new color version	PC	\$279
Greenleaf Data Windows	PC	\$155
w/source	PC	\$269
TurboWINDOW/C - for Turbo C	PC	\$ 79
Windows for C - fast	PC	\$149
Windows for Data - validation	PC	\$239
Vitamin C - screen I/O	PC	\$159
View Manager - by Blaise	PC	\$199
ZView - screen generator	MS	\$129

Atari ST & Amiga

We carry full lines of Manx & Lattice.

DBASE Language

Clipper compiler	PC	\$399
dBASE II	MS	\$329
dBASE III Plus	PC	\$429

Call for a catalog, literature and solid value

800-421-8006

THE PROGRAMMER'S SHOP
Your complete source for software, services and answers

5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 10/87

RECENT DISCOVERY

COBOL/2 by MicroFocus - OS/2 and DOS Protected mode. 9 language variants, network support, syntax checker, fast code, debugger, huge model, interface to C. PC \$749

DBASE Language Cont.

dBASE III LANPack	PC	\$649
DBXL Interpreter by Word Tech	PC	\$139
FoxBASE+ - single user	MS	\$349
Quicksilver by Word Tech	PC	\$439

DBASE Support

dAnalyst	PC	\$ 89
dBase Tools for C	PC	\$ 65
dBrief with Brief	PC	Call
DBC ISAM by Lattice	MS	\$169
Documentor - dFlow superset	MS	\$229
Genifer by Bytel-code generator	MS	\$279
QuickCode III Plus	MS	\$239
R&R Report Generator	MS	\$139
Seck-It - Query-by-example	PC	\$ 79
Silver Comm Library	MS	\$139
Tom Rettig's Library	PC	\$ 79
UI Programmer - user interfaces	PC	\$249

Fortran & Supporting

50:More FORTRAN	PC	\$ 95
Fortran Addenda	PC	\$139
I/O Pro - screen development	PC	\$129
MS Fortran - 4.0, full '77	MS	\$269
No Limit - Fortran Scientific	PC	\$109
PC-Fortran Tools - xref, pprint	PC	\$165
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	\$129

Multilanguage Support

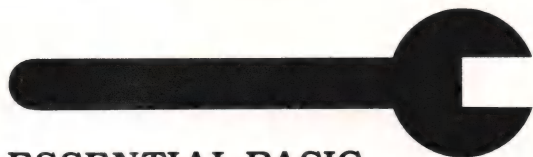
BTRIEVE ISAM	MS	\$185
BTRIEVE/N-multiuser	MS	\$455
GSS Graphics Dev't Toolkit	PC	\$375
HALO Development Package	MS	\$389
Graphics	PS	\$209
Hoops Graphics Library	PC	\$549
Informix 4GL-application builder	PC	\$789
Informix SQL - ANSI standard	PC	\$639
NET-TOOLS - NET-BIOS	PC	\$129
Opt Tech Sort - sort, merge	MS	\$ 99
PANEL	MS	\$215
Panel Plus	MS	\$395
Pfinish - by Phoenix	MS	\$229
Polyboost - speed I/O, keyboard	PC	Call
Prime Factor FFT - 8087/287	PC	\$145
PVCS Corporate or Personal	MS	Call
Report Option - for Xtrieve	MS	\$109
Screen Sculptor	PC	\$ 89
SSP/PC - 145+ math routines	PC	\$269
Synergy - create user interfaces	MS	\$375
Xtrieve - organize database	MS	\$199
ZAP Communications - VT 100	PC	\$ 89

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add \$3 item.

Ask for a **FREE REPORT**: "Interviews with Authors & Users:
How Can These Products Raise Productivity and Help You Write Better Programs?" from

THE PROGRAMMER'S SHOP

EXIM TOOLKIT:



ESSENTIAL BASIC PROGRAMMING TOOLS

A **must** for all BASIC programmers who use Microsoft's QuickBASIC or IBM's PC BASIC compilers, our feature-rich, easy-to-use **EXIM Toolkit** provides dozens of useful routines unavailable in BASIC or any other single software library sold today!

Here's just some of what the **EXIM Toolkit** can do:

With our **Screen Management** module, you get features like four-way scrolling, windowing and data entry/display in screen forms. You can also write to screen many times faster than with BASIC and develop software to copy areas of the screen for cut and paste functions.

The multi-user **Data Base Management** module lets multiple users access the same data files across a local area network and still maintain data integrity. Users can build, add and delete indexes; reformat files and much more. Now BASIC users can control data as never before.

The **Network Interface modules** make it possible for BASIC programs to issue NETBIOS commands.

BASIC limits you to 64K of RAM for strings. **EXIM Toolkit's Memory Manager** lets you access up to **256K additional RAM** for strings. And, we offer free technical support, 7 days a week.

Order your **EXIM Toolkit** today. It's only \$99.95, (\$199.95 with Network Interface Modules) plus \$5.00 shipping and handling. (No licensing or royalties required.) If within 30 days, you are not completely satisfied with the **EXIM Toolkit**, return it to your dealer for a complete refund.

The **EXIM Toolkit** is so comprehensive, we're convinced it's the only BASIC Software Library you'll ever need.

PS: \$ 45



EXIM SERVICES OF N.A. INC.

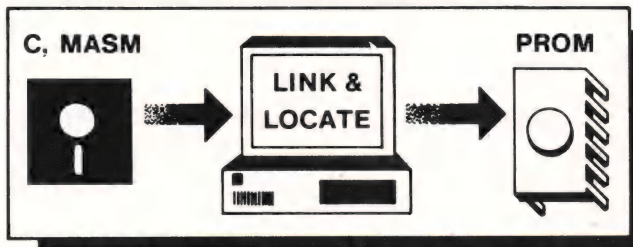
P.O. Box 5417

Clinton, New Jersey 08809

(201) 735-7640

CIRCLE 172 ON READER SERVICE CARD

FIRMWARE DEVELOPMENT



LINK & LOCATE enables PC users to produce ROM-based firmware for 8086/87/186 from object files generated by popular C compilers, such as from Microsoft, Lattice and Borland's Turbo C, and MASM from Microsoft. Provides full control of segment placement anywhere in memory. Supports output of Intel HEX file for PROM programmers, Intel OMF absolute object file for symbolic debuggers and in-circuit emulators. Includes Intel compatible linker, locator, librarian, hex formatter and cross reference generator. \$350.

NEW! Includes utility to support PC based PROM programmers.

ESI SYSTEMS & SOFTWARE

PS: \$329

3303 Harbor Blvd., C11, Costa Mesa, CA 92626

Phone (714) 241-8650 FAX (714) 241-0377 TWX 910-695-0125

CIRCLE 173 ON READER SERVICE CARD

Now you see it; Now you don't!

TSR's made easy with /*resident_C*/

Finally! You write your C program and we make it resident. No strings attached. Want to process interrupts also? No problem. /*resident_C*/ is a set of C functions that enable you to process interrupts and/or make your programs "terminate stay resident" like Sidekick.

We've done the research, testing, and grunt work to make your TSR programs safe, compatible and easy. Be aware of the "ifs, ands and buts" when writing interrupt handling software. Only /*resident_C*/ has all that you need to write reliable interrupt handlers without the worries.

We don't ask you to trust us either. Our manual explains what we are doing very clearly and our documented source code is available. In addition, our working demos give you clear examples of resident programs, interrupt handlers and resident libraries. There is no other product that can do what /*resident_C*/ can do for you.

/* resident_C */ is the perfect complement to the other library products available from Essential Software.



List: \$99 / with Source add \$99

PS: \$79 / with Source add \$79

CIRCLE 174 ON READER SERVICE CARD

Call or Circle Reader Service Number for Your **FREE Study**.

HOURS: 8:30 A.M.-8:00 P.M. E.S.T.
5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510

800-421-8006

THE PROGRAMMER'S SHOP

Your complete source for software, services and answers

INTEGERS DON'T FLOAT

Listing One (Text begins on page 48.)

Listing 1

```

;      ISQRT32.ASM - 32 bit integer test program for no 8087
;      By Ray Mariella, March 87
;      page    ,96
;
crlf    macro
        mov     dl,13
        call    char_out
        mov     dl,10
        call    char_out
        endm

;time_print macro byte_var, byte_string
        local plenty
        mov     di,byte_string          ;output a colon or period
        call    char_out
        cmp     byte_var,9
        ja     plenty
        mov     dl,'0'
        call    char_out                ;space holder if var<10
plenty: mov     al,byte_var              ;minutes, secs, or hnds
        xor     ah,ah
        call    dec_out
        endm

;
;
data     segment word public 'DATA'
base     dw 10                          ;base for dec_out
uper     db ?                          ;for time_print routine
secs     db ?
hnds     db ?
announc   db ' 60000  32 bit square roots ',13,10,'$'
data     ends

;
;
stack    segment stack
        dw 64 dup(?)
stack     ends
;
;
code     segment word public 'code'
        assume cs:code, ds:data, ss:stack
;
sqrt:    mov     ax,data
        mov     ds,ax
crlf     mov     dx,offset announc
        mov     ah,9                    ;print string function
        int     21h                    ;DOS interrupt
        mov     dl,13
        call    char_out
;
herald:   crlf
;
rolling:  xor     di,di                  ;upper 16
        mov     si,32767                ;lower 16
;
goodies:  call    update
;
;      square root procedure of DI:SI via 8086,
;
start:    mov     bx,1                  ;initial value for infimum
        mov     dx,di                  ;initial supremum, upper 16
        mov     ax,si                  ;initial supremum, lower 16
;
biggest:  or      dx,dx                  ;test if upper 16 =0 yet
        jz      words                  ;if yes, we don't need upper 16 now
        rcr     dx,1                    ;supr. upper 16/2
        rcr     ax,1                    ;supr. lower 16/2 + carry from upper
        shl     bx,1                    ;infim.*2
        jmp     short biggest
;
words:    or      bx,bx                  ;now infim. and supr. are 16 bits
        jnz     checkem                ;if BX was made 0, correct it!
        mov     bx,0ffffh              ;if not, O.K. to continue
        ;correction for the largest 32 bitters
checkem:  mov     dx,ax                  ;supr. in ax,dx
        mov     cx,bx                  ;infim. in bx,cx
        ;
logit:    shl     cx,1                  ;infimum*2
        jc      average                ;necessary for large integers
        cmp     cx,dx                  ;infimum*2 > supremum?
        jae     average                ;if so, ready to average
        shr     dx,1                    ;if not, supr/2
        mov     ax,dx                  ;store latest values
        mov     bx,cx
        jmp     short logit
;
average:  ;ready for averaging
        add     bx,ax                  ;(infim.+ supr.)
        rcr     bx,1                  ;average value for first guess
;
Newton:   REPT 2
        mov     ax,si                  ;lower 16 of target in ax,
        mov     dx,di                  ;upper 16 of in dx, for division
        cmp     bx,dx                  ;this is for near FFFF:0000 and up
        jz      cont                  ;but not needed for FFFD:0000 and less
        div     bx                     ;N/(g1) in AX, now get g2
        add     bx,ax                  ;Newton's method  g2 = (g1 +N/g1)/2
        rcr     bx,1                  ;bx now has g2
        endm
;
cont:     inc     di
        cmp     di, 60000
        ja     quit
        jmp     start
;
quit:     call    update

```

(continued on page 100)

New!

386|DEBUG

- A symbolic debugger for 80386 32-bit *protected mode* programs which run under Phar Lap's 386|DOS-Extender™
- Breakpoints, data watchpoints, and built-in disassembler
- Fully compatible with Phar Lap's 386|ASM/LINK, the MetaWare 80386 High C™ and Professional Pascal™ compilers, and the Green Hills 80386 Fortran compiler
- Runs on all DOS-based PCs equipped with an 80386 CPU, including the Compaq® DESKPRO 386™, the IBM®PS/2™ Model 80, and most accelerator cards, including the Intel Inboard™ 386/AT
- \$195—Available today

(617) 661-1510

Phar Lap Software, Inc.
60 Aberdeen Ave.
Cambridge, MA 02138



"The 80386 Software Experts"

CIRCLE 176 ON READER SERVICE CARD

Would you like copy protection and customer satisfaction?

Here's a better way to protect your software.
It's called the Secom Key, and it works.

The Key is completely transparent to the end user.

Won't interfere with peripheral operations.

Doesn't occupy the disk drive.

The Key allows **unlimited backup copies**.

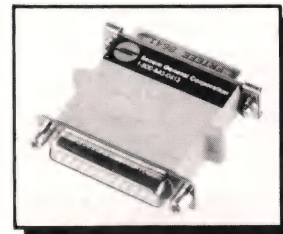
Makes site licensing easy and auditable.

Easily installed. Uses only 1000 bytes.

Over 100,000 have been sold worldwide.

Same size as RS-232 plug.

Available in quantities for as low as \$19.95.



The Secom Key...
for real
software
protection.

For more information, contact

Secom Information Products Company
A Subsidiary of Secom General Corporation

Call Toll-free 1-800-843-0413

500 Franklin Square 1829 East Franklin Street Chapel Hill, NC 27514

CIRCLE 177 ON READER SERVICE CARD

ADD TO THE POWER OF YOUR PROGRAMS WHILE YOU SAVE TIME AND MONEY!

CBTREE does it all! Your best value in a B+tree source!

Save programming time and effort.

You can develop exciting file access programs quickly and easily because CBTREE provides a simple but powerful program interface to all B+tree operations. Every aspect of CBTREE is covered thoroughly in the 70 page Users Manual with complete examples. Sample programs are provided on disk.

Gain flexibility in designing your applications.

CBTREE lets you use multiple keys, variable key lengths, concatenated keys, and any data record size and record length. You can customize the B+tree parameters using utilities provided.

Your programs will be using the most efficient searching techniques.

B+trees use efficient search techniques that require fewer disk seeks than other methods. CBTREE guarantees an optimized maximum search path and always remains balanced. CBTREE is optimized for speed. You will be using a commercial quality, reliable and powerful tool. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

Access any record or group of records by:

- its absolute position in the index (GETFRST and GETLAST),
- its relative location in the index (GETPRV, GETNXT and GETSEQ),
- an exact match to a key (GETREC),
- a partial match to a key (GETPAR, GETALL and GETKEYS)
- a lexical relation to a key (GETLT, GETLE, GETGT and GETGE).
- You may also add, delete and update any record without the need to reorganize the index (INSERT, ISRTKY, DELETE, DELTKY and NEWLOC).
- Block retrieval calls speed up sequential processing.

Increase your implementation productivity.

CBTREE is over 6,000 lines of tightly written, commented C source code. The driver module is only 20K and links into your programs.

Port your applications to other machine environments.

The C source code that you receive can be compiled on all popular C compilers for the IBM PC and also under Unix, Xenix, and AmigaDos! No royalties on your applications that use CBTREE. CBTREE supports multi-user and network applications.

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!

CBTREE compares favorably with other software selling at 2,3 and 4 times our price.

Sold on unconditional money-back guarantee.

YOU PAY ONLY \$99.00 - A MONEY-SAVING PRICE!

TO ORDER OR FOR ADDITIONAL INFORMATION

CALL (703) 356-7029 or (703) 847-1743

OR WRITE



Peacock Systems, Inc., 2108-C Gallows Road, Vienna, VA 22180

CIRCLE 178 ON READER SERVICE CARD

INTEGERS DON'T FLOAT

Listing One *(Listing continued, text begins on page 48.)*

```

        xor     al,al
        mov     ah,4Ch
        int     21h
;
;       output a hex word in decimal
;
;       CX,AX,DX destroyed
;
dec_out  proc     near
        xor     cx,cx
another: inc     cx
        xor     dx,dx
        div     base          ;base is 10 decimal!
        push    dx            ;remainder is less sig digits
        or      ax,ax         ;is the quotient zero?
        jnz     another       ;if not, more number to convert
print_dig:
        pop     dx            ;retrive digit from stack
        add     dl,'0'        ;ascii offset
        call    char_out
        loop    print_dig     ;do all of the digits
        ret
dec_out  endp
;
;       output a single character
;
char_out proc     near
        mov     ah,2          ;output char function
        int     21h          ;do it
        ret
char_out endp
;
;
update  proc     near
        mov     ah,2Ch        ;get dos time
        int     21h          ;hour in ch, mins in cl,secs in dh
        mov     upper,cl
        mov     secs,dh
        mov     hnds,dl
        mov     al,ch
        xor     ah,ah
        call    dec_out
;
;       time_print upper,': '
;
;       time_print secs,': '
;
;       time_print hnds,': '
;
        crlf
        ret
update  endp
;
code    ends
end      sqrt

```

End Listing One

Listing Two

Listing 2

```

;
;       R32COMP.ASM - 32 bit sqrt , compares CPU, NDP
;       By Ray Mariella, 30 March 87 - increments the upper 16 bits
;                                     0000:7FFF to FFFF:7FFF
;       requires 8087 or 80287
;       page      ,96
.8087
;
;
crlf    macro
        mov     dl,13
        call    char_out
        mov     dl,10
        call    char_out
        endm
;
time_print macro byte_var, byte_string
        local plenty
        mov     dl,byte_string          ;output a colon or period
        call    char_out
        cmp     byte_var,9
        ja      plenty
        mov     dl,'0'
        call    char_out
plenty: mov     al,byte_var
        xor     ah,ah
        call    dec_out
        endm
;
;
data    segment public 'DATA'
        even
base     dw      10          ;base to print the numbers in
BIGGUN  dq      ?
rootp   dd      ?
upper   db      ?
secs    db      ?
hnds    db      ?
announc db      ?
data    ends
;
;
stack   segment stack
        dw      64 dup(?)
stack   ends
;
;
code    segment public 'CODE'
        assume cs:code, ds:data, ss:stack
;
sqrt:   push    bp

```



```

mov ax,data
mov ds,ax
mov bp,offset biggun

crlf:
mov dx,offset announc
mov ah,9
int 21h ;print string function
;DOS interrupt

;
mov si,32767
xor di,di
goodies: call update
;
; square root procedure via 8086, DI:SI
CPU: mov bx,1 ;guess1
mov dx,di ;guess2
mov ax,si
or dx,dx
jz words
rcr dx,1 ;upper 16/2
rcr ax,1 ;lower 16/2 + carry from upper
shl bx,1 ;guess1*2
jmp short biggest

words: or bx,bx ;next is for guess1 and guess2 16 bits
jnz checkm
mov bx,65535 ;if all 32 were used, CF is set
;in case all 32 bits were used

; checkm: mov dx,ax ;guess2 ax,dx
mov cx,bx ;guess1 bx,cx

; logit: shl cx,1 ;guess1
jnc average ;necessary for very large integers
cmp cx,dx ;larger than guess2?
jae average ;if not, guess2/2
shr dx,1
mov ax,dx
mov bx,cx
jmp short logit

average: ;ready for averaging
add bx,ax
rcr bx,1 ;average value

Newton: REPT 2
mov ax,si ;lower 16
mov dx,di ;prepare for division, upper 16 in dx
cmp bx,dx ;needed for really BIG ints
je quit
div bx
add bx,ax ;ax still has target, bx first guess
rcr bx,1 ;newton
endm

;
quit: inc di
jz done
jmp CPU

done: call update
;
; square root via 8087
; if you need roots of 7FFF:FFFF and less, BIGGUN can be 32 bits,
; and ROOTP can be a 16 bits. The extra length is needed here because
; the 8087 does not expect unsigned integers.
;
xor di,di ;8087 loads from memory,
mov ds:[bp],si ;not regs directly
NDP: mov ds:[bp+2],di
fild biggun ;put integer into 8087 stack
fsqrt
fstp rootp ;store to memory, too
fwait

;
; we now have an 8087 square root -rootp
;
inc di
jnz NDP

call update
pop bp
xor al,al
mov ah,4Ch
int 21h

;
; output a hex word in decimal
;
; CX,AX,DX destroyed
;
dec_out: proc near
xor cx,cx
another: inc cx
dx,dx
div base ;base is 10 decimal!
push dx ;remainder is less sig digits
or ax,ax ;is the quotient zero?
jnz another ;if not, more number to convert

print_dig: pop dx ;retrieve digit from stack
add dl,'0' ;ascii offset
call char_out
loop print_dig ;do all of the digits
ret

dec_out: endp

;
; output a single character
;

```

(continued on next page)

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"

"... a technological tour de
force for fast PC graphics."

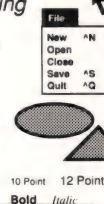
NO ROYALTIES!

MetaWINDOW is a
NEW! - QuickWINDOW/C
All the features of MetaWINDOW
for Microsoft Quick/C! - \$95*
and window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded
set of graphic drawing functions,
plus the added functionality and
performance required for designing
multi-window desktop applications.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- multiple fonts



Enhanced Features!

- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, under-line or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports IBM's new PS/2 VGA and MCGA graphics.

MetaWINDOW comes complete with
language bindings for 20 popular C,
Pascal and Fortran compilers, plus
dynamic runtime support for over 50
graphics adaptors and input devices.

MetaWINDOW

Advanced Graphics Toolkit

4 disks, 3 260 page manuals - \$195*

NEW! - TurboWINDOW/Pascal

All the features of MetaWINDOW for
Borland Turbo Pascal Ver. 4! - \$95*
* Plus \$5.00 shipping and handling

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550



METAGRAPHS
SOFTWARE CORPORATION

269 Mount Hermon Road
Scotts Valley, CA 95066

INTEGERS DON'T FLOAT

Listing Two (Listing continued, text begins on page 48.)

```

char_out proc near
    mov     ah,2             ;output char function
    int     21h             ;do it
    ret
char_out endp
;
;
update proc near
    mov     ah,2ch           ;get dos time
    int     21h             ;hour in ch, mins in cl,secs in dh
    mov     uper,cl
    mov     secs,dh
    mov     hnds,dl
    mov     al,ch
    xor     ah,ah
    call    dec_out
;
;    time_print uper,':'
;
;    time_print secs,':'
;
;    time_print hnds,','
;
;    crlf
;    ret
update endp
;
code ends
end        sqrt

```

End Listing Two

Listing Three

Listing 3

```

; R32fail.ASM - 32 bit integer test program REPT macros
; looks for the first time that Newton fails 0,-1
; By Ray Mariella, April 87
; page ,96
.8087
;
; crlf macro
; mov     dl,13
; call    char_out
; mov     dl,10
; call    char_out
; endm
;
; time_print macro byte_var, byte_string
; local plenty
; mov     dl,byte_string           ;output a colon or period
; call    char_out
; cmp     byte_var,9
; ja      plenty
; mov     dl,'0'
; call    char_out
; space holder if var<10
; plenty: mov     al,byte_var
; minutes, secs, or hnds
; xor     ah,ah
; call    dec_out
; endm
;
; data segment word public 'DATA'
; base dw 10
; BIGGUN dq ?
; rootp dd ?
; contwd dw ?
; uper db ?
; secs db ?
; hnds db ?
; announc db 'incr. lower 16 from 1 ',13,10,'$'
; intermed db ' passed 65535 ',13,10,'$'
; data ends
;
; stack segment stack
; dw 64 dup(?)
; stack ends
;
; code segment word public 'code'
; assume cs:code, ds:data, ss:stack
;
; sqrt: push     bp
; mov     ax,data
; mov     ds,ax
; mov     bp,offset biggun
; fclx
; fstcw contwd
; and contwd,1111001111111111B ;clear 8087 exceptions, if any
; fldcw contwd ;get control word
; round to nearest
; load changed control word
; crlf
; mov     dx,offset announc
; mov     ah,9
; int     21h ;print string function
; mov     dl,13
; call    char_out ;DOS interrupt
;
; herald: crlf
;
; rolling: xor     bx,bx
; xor     ax,ax
; mov     si,1
; mov     ds:[bp],si ;lower 16, will vary
; mov     di,0 ;BIGGUN lower 16
; mov     ds:[bp+2],di ;upper 16
;
;
;

```



```

goodies: call update
          crlf
start:    FILD BIGGUN
;
; square root procedure via 8086/V30
;
          mov     bx,1           ;guess1
          mov     dx,di         ;guess2
          mov     ax,si
;
          REPT 8
          or      dx,dx
          jz      halfway
          rcr     dx,1           ;upper 16/2
          rcr     ax,1           ;lower 16/2 + carry from upper
          shl     bx,1           ;guess1*2
          endm
          jmp     short rest
halfway:  jmp     short words
;
rest:
          REPT 8
          or      dx,dx
          jz      words
          rcr     dx,1           ;upper 16/2
          rcr     ax,1           ;lower 16/2 + carry from upper
          shl     bx,1           ;guess1*2
          endm
          ;next is for guess1 and guess2 16 bits
words:    FSQRT
          or      bx,bx
          jnz     checkem        ;if all 32 were used, CF is set
          mov     bx,65535       ;in case all 32 bits were used
;
checkem:  mov     dx,ax           ;guess2 ax,dx
          mov     cx,bx           ;guess1 bx,cx
;
logit:
          REPT 8
          shl     cx,1           ;guess1
          jc      average        ;necessary for 2000:4000 and up
          cmp     cx,dx           ;larger than guess2?
          jae     average
          shr     dx,1           ;if not, guess2/2
          mov     ax,dx
          mov     bx,cx
          endm
          ;ready for averaging
average:  FISTP  rootp
          add     bx,ax
          rcr     bx,1           ;average value
;
Newton:
          REPT 2
          mov     ax,si           ;lower 16
          mov     dx,di           ;prepare for division, upper 16 in dx
          cmp     bx,dx           ;for FFFE:0000 and up
          je      quit
          div     bx             ;ax still has target, bx first guess
          add     bx,ax           ;newton
          rcr     bx,1
          endm
          FWAIT
done:     mov     ax,bx
          mov     dx,word ptr ds:[bp+8] ;ax, and bx have approx. root
          xor     ax,dx           ;bp+8 is rootp, 8087 root
          jz      cont           ;see if rootp agrees
          cmp     bx,dx
          ja      quit
;
belo:     inc     bx
          cmp     bx,dx
          jnz     quit
;
cont:     inc     si               ;lower 16
          jnz     notyet
          inc     di
          mov     ax,di
          call    dec_out
          mov     dx,offset intermed
          mov     ah,9
          int     21h
          crlf
notyet:   mov     ds:[bp+2],di     ;biggun upper 16
          mov     ds:[bp],si      ;biggun lower 16
          jmp     start
;
quit:     call    update
          mov     ax,di
          call    dec_out
          mov     dl,' '
          call    char_out
          mov     ax,si
          call    dec_out
          crlf
          pop     bp
          xor     al,al
          mov     ah,4Ch
          int     21h
          ret
;
; output a hex word in decimal
;
; CX,AX,DX destroyed
;
dec_out  proc  near
another:  xor     cx,cx
          inc     cx

```

(continued on next page)

The Fast Cure For A Slowing Hard Disk



"Vopt is something of a miracle. It performs its disk reorganization chores in seconds, instead of the minutes and even hours some other utilities can take.

...a bargain, Vopt is fast, safe, effective, and even fun to use. What more could you want?"



**Glenn Hart, PC Magazine
May 12, 1987, Page 36.**

"The overall efficiency of my computer system was significantly improved."

**William G. Harrington,
The National Law Journal
June 29, 1987, Page 14.**

Vopt gives you faster hard disk access in seconds!

When DOS creates a file, it scatters file fragments over the disk surfaces. It takes time to collect those fragments when you need the data, so your system runs slower and slower as your files grow more fragmented.

Vopt organizes your files the way DOS should have written them--contiguously--so file retrieval is easy and fast!

\$49.95 \$3 shipping/handling.
CA add 6% sales tax.

GOLDEN BOW SYSTEMS



**2870 Fifth Avenue
Suite 201
San Diego, CA 92103
619/298-9349**

Vopt operates with DOS systems, including
PS/2, with 512Kb RAM.

Vopt is a trademark of Golden Bow Systems.

CIRCLE 180 ON READER SERVICE CARD

INTEGERS DON'T FLOAT

Listing Three (Listing continued, text begins on page 48.)

```

        xor     dx,dx
        div     base                ;base is 10 decimal!
        push    dx                  ;remainder is less sig digits
        or      ax,ax              ;is the quotient zero?
        jnz     another            ;if not, more number to convert

print_dig:
        pop     dx                  ;retrive digit from stack
        add     dl,'0'              ;ascii offset
        call    char_out
        loop    print_dig          ;do all of the digits
        ret
dec_out  endp
;
;      output a single character from dl
;
char_out proc    near
        mov     ah,2                ;output char function
        int     21h                ;do it
        ret
char_out endp
;
;
update  proc    near
        mov     ah,2ch              ;get dos time
        int     21h                ;hour in ch, mins in cl,secs in dh
        mov     uper,cl
        mov     secs,dh
        mov     hnds,dl
        mov     al,ch
        xor     ah,ah
        call    dec_out
;
;      time_print uper,':'
;
;      time_print secs,':'
;
;      time_print hnds,','
;
        crlf
        ret
update  endp
;
code    ends
end      sqrt

```

End Listing Three

Listing Four

```

Listing 4
;
;      RALL16 - square roots of 1 to 65535
;      By Ray Marilella, Nov 1986
;      page    ,96
;
crlf    macro
        mov     dl,13
        call    char_out
        mov     dl,10
        call    char_out
        endm
;
time_print macro byte_var, byte_string
        local plenty
        mov     dl,byte_string      ;output a colon or period
        call    char_out
        cmp     byte_var,9
        ja      plenty
        mov     dl,'0'              ;space holder if var<10
        call    char_out
plenty:  mov     al,byte_var          ;minutes, secs, or hnds
        xor     ah,ah
        call    dec_out
        endm
;
;
data    segment    word public 'DATA'
        even
base     dw      10                  ;base to print the numbers in
uper     db      ?
secs     db      ?
hnds     db      ?
announc  db      'square roots of 1 - 65535 ',13,10,'$'
data     ends
;
;
stack   segment    stack
        dw      64 dup(?)
stack   ends
;
;
code     segment    word public 'CODE'
        assume cs:code, ds:data, ss:stack
        even
;
sqrt:    mov     ax,data
        mov     ds,ax
crlf     mov     dx,offset announc
        mov     ah,9                ;print string function
        int     21h                ;DOS interrupt
        mov     dl,13
        call    char_out
;

```


MS-DOS Enhancements and Unix-Like Features for MS-DOS



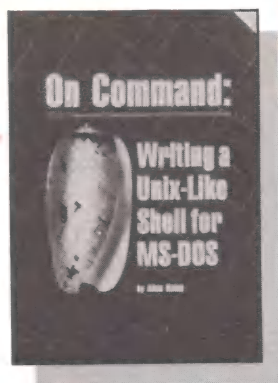
Taming MS-DOS

by Thom Hogan

Taming MS-DOS takes you beyond the basics, picking up where your MS-DOS manual leaves off. You'll learn how to maximize your batch files with routines using redirection, filters and pipes, and routines that prevent accidental reformatting of your hard disk, redefine function keys, and locate files within subdirectories. You'll also learn how to create configurable AUTOEXEC.BAT files, and how to customize CONFIG.SYS and use ANSI.SYS to change the appearance of MS-DOS. **Taming MS-DOS** includes ready-to-use assembly language programs that enhance MS-DOS.

The programs, including batch files and MS-DOS enhancements, are available on disk with full source code.

Book & Disk (MS-DOS) Item #59-3 \$34.95
Book Item #24-0 \$19.95



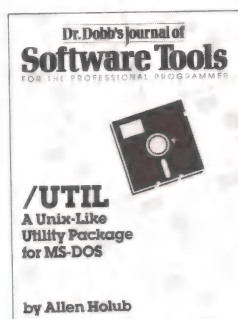
On Command: Writing a Unix-Like Shell for MS-DOS

by Allen Holub

This book and ready-to-use program demonstrate how to write a Unix-like shell for MS-DOS, with techniques applicable to most other programming languages as well. The book and disk include a detailed description and working version of the Shell, complete C source code, a thorough discussion of low-level MS-DOS interfacing, and significant examples of C programming at the system level. Supported features include: read, aliases, history, redirection and pipes, Unix-like command syntax, MS-DOS-compatible prompt support, C-Shell-based shell scripts, and a Shell variable that expands to the contents of a file so a program can produce text that is used by Shell scripts. The Unix-like control flow includes: if/then/else; while; foreach; switch/case; break; continue.

The ready-to-use program and all C source code are included on disk. For IBM PC and direct compatibles.

Book & Disk (MS-DOS) Item #29-1 \$39.95

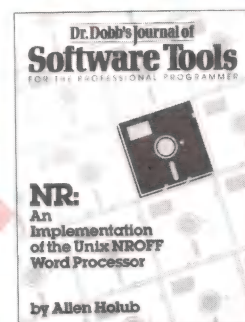


Util

by Allen Holub

When used with the Shell, this collection of utility programs and subroutines provides you with a fully functional subset of the Unix environment. Many of the utilities may also be used independently. You'll find executable versions of cat; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printevn; rm; rmdir; sub; and chmod. **Util** includes complete source code on disk and all programs (and most of the utility subroutines) are fully documented in a Unix-style manual. For IBM PC and direct compatibles.

Manual & Disk (MS-DOS) Item #12-7 \$29.95



NR: An Implementation of the Unix NROFF Word Processor

by Allen Holub

NR is a text formatter that is written in C and compatible with UNIX's NROFF. Complete source code is included in the **NR** package so that it can be easily customized to fit your needs. **NR** also includes an implementation of the -ms macro package and an in-depth description of how -ms works. **NR** does hyphenation and simple proportional spacing. It supports automatic table of contents and index generation, automatic footnotes and endnotes, italics, boldface, and more. **NR** also contains: extensive macro and string capability; number registers in various formats; diversions and diversion traps; input and output line traps.

NR is easily configurable for most printers. Both the ready-to-use program and full source code are included. For PC compatibles.
Manual & Disk (MS-DOS) Item #33-X \$29.95

OUR GIFT TO YOU!

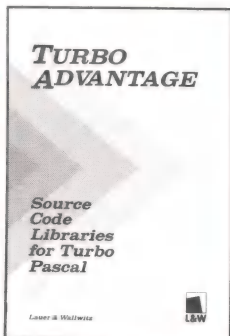


Receive a FREE MUG When You Order This Special Holiday Package!

Save 15%
 Receive **On Command**, **Util** and **NR** together for only \$85.95!
 You get all the convenience of Unix-like features and save 15%.

Unix-like Features Package Item #167 \$85.95

Turbo Pascal Tools



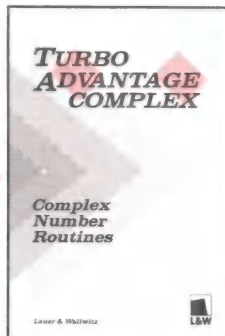
TURBO Advantage: Source Code Libraries for Turbo Pascal

by Lauer & Wallwitz
The **TURBO Advantage** library contains more than 220 routines complete with source code, sample programs and documentation. Routines are organized and documented under the following categories: bit manipulation; file management; MS-DOS support; sorting routines and spline integration; string operations; and more!

A detailed manual includes a description of each routine, an explanation of the methods used, the calling sequence, and a simple example. Source code is included. For MS-DOS systems.

Manual & Disk (MS-DOS)
Item #26-7

\$49.95



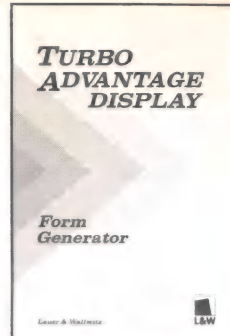
TURBO Advantage Complex: Complex Number Routines for Turbo Pascal

by Lauer & Wallwitz
TURBO Advantage Complex provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations, the size of arrays are easily adapted. Each type declaration is a record with both a real and an imaginary part. Use these procedures to build more sophisticated functions in your own programs. **TURBO Advantage Complex** also demonstrates the use of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; and more.

Source code and documentation included. For MS-DOS systems. Some of the **TURBO Advantage Complex** routines are most effectively used with routines contained in **TURBO Advantage**.

Manual & Disk (MS-DOS)
Item #27-5

\$89.95

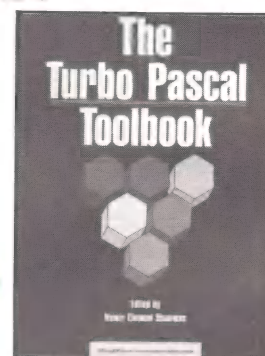


TURBO Advantage Display: Form Generator for Turbo Pascal

by Lauer & Wallwitz
The **TURBO Advantage Display** form generator includes an easy-to-use form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, full source code, and documentation. For MS-DOS systems. Some of the **TURBO Advantage** routines are necessary to compile **TURBO Advantage Display**.

Manual & Disk (MS-DOS)
Item #28-3

\$69.95



The Turbo Pascal Toolbook Edited by

Namir Clement Shammass
Featuring over 800K of fully documented Turbo Pascal source code, **The Turbo Pascal Toolbook** includes: an extensive library of low-level routines; external sorting and searching tools, presenting a new database routine that combines the best features of the B-, B+, and B++ trees; window management tips; and much more. All routines, libraries, and sample programs are on disk for MS-DOS systems.

Book & Disk (MS-DOS)
Item #61-5
Book only
Item #25-9

\$45.95

\$25.95

OUR GIFT TO YOU!



Receive a FREE MUG
When Your Order
This Special Holiday
Package!

Receive **TURBO Advantage**
together with
TURBO Advantage Complex
or **TURBO Advantage Display**
and **SAVE 20%**

TURBO Advantage & TURBO Advantage Complex
Item #070A **\$115**
TURBO Advantage & TURBO Advantage Display
Item #070B **\$99.95**

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 871 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

M&T Books

501 Galveston Dr.
Redwood City, CA 94063

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES


```

        call update
        mov     cx,65535
;
; square root procedure via 8086, integer in CX
boundit: mov     ax,1           ;infimum - will be lower bound
        mov     dx,cx         ;supremum - will be upper bound
;
; the next section gets the max lower bound and the min upper bound
        REPT 8
        shl     ax,1           ;mpy infimum by 2
        cmp     ax,dx         ;above supremum ?
        ja      root          ; then done
        shr     dx,1           ;if not, div supremum by 2
        mov     bx,dx         ;store supr.
        mov     si,ax         ;store infim.
        ENDM
;
root:    add     bx,si          ;get avg. of bounds for
        shr     bx,1           ;first guess of root
        mov     ax,cx
;
; Newton's Method => X = (Xo + N/Xo)/2
;
Newton:  xor     dx,dx         ;prepare for division
        div     bx            ;ax still has target
        add     bx,ax         ;newton
        shr     bx,1
;
; we now have a square root in bx
;
; to print, remove the next 8 semicolons
;
        mov     ax,cx
        call    dec_out
        mov     dl,' '
        call    char_out
        mov     ax,bx
        call    dec_out
        crlf
didit:   loop boundit
;
done:    call update
        crlf
        xor     al,al
        mov     ah,4Ch
        int     21h
;
; output a hex word in decimal
;
; CX,AX,DX destroyed
dec_out  proc    near
        xor     cx,cx
        inc     cx
        xor     dx,dx
        div     base          ;base is 10 decimal!
        push    dx            ;remainder is less sig digits
        or      ax,ax         ;is the quotient zero?
        jnz     another       ;if not, more number to convert
print_dig: pop     dx          ;retrive digit from stack
        add     dl,'0'        ;ascii offset
        call    char_out
        loop    print_dig     ;do all of the digits
        ret
dec_out  endp
;
; output a single character
;
char_out proc    near
        mov     ah,2
        int     21h           ;output char function
        ret       ;do it
char_out endp
;
;
update  proc    near
        mov     ah,2ch         ;get dos time
        int     21h
        mov     uper,cl        ;hour in ch, mins in cl,secs in dh
        mov     secs,dh
        mov     hnds,dl
        mov     al,ch
        xor     ah,ah
        call    dec_out
;
        time_print uper,':'
;
        time_print secs,':'
;
        time_print hnds,','
;
        crlf
        ret
update  endp
;
code    ends
        end      sqrt

```

End Listings



SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

\$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM
INDEPENDENT

**MACHINE
INDEPENDENT
SOFTWARE
CORPORATION**

1415 NORTHGATE SQUARE #21D
RESTON, VA 22090

VISA/Master Charge accepted
(703) 435-0413

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.
MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL logo are trademarks of
Kurtzberg Computer Systems.

CIRCLE 181 ON READER SERVICE CARD

Listing One (Text begins on page 54.)

```
Listing 1

/* POPWIN.I: Routines for pop-up windows */
/* Externally defined are gotoxy(), wrtcha(), window(), chattr(), */
/* and videomode(). */
/* Notes: Programmer is responsible for assuring that the pop-up */
/* defined in the POPDESCR structure is large enough to */
/* contain the text. */
/* Save your cursor position before calling these routines. */
/* They don't preserve or restore caller's cursor. */
/* Unlike menubar, the POPDESCR structure doesn't contain a */
/* pointer to the text elements. Thus, you can use the */
/* routines to create dialog boxes and help windows. */
/* ----- */
typedef struct { /* window descriptor/control structure */
    int left, top, right, bottom; /* window location, inclusive */
    char textAttr; /* text attribute in window */
    int border; /* 0 = none, 1 = single, 2 = double */
} POPDESCR;

static bord[16] = { /* border characters */
    { 196, 179, 218, 191, 217, 192 },
    { 205, 186, 201, 187, 188, 200 }
};

void popMake (POPDESCR *win) /* create pop-up window */
/* Must initialize the POPDESCR structure before calling. */
/* Does not write text to the window, but only creates it. After */
/* this fn returns, you can write text using popPuts() and */
/* control the cursor with popxy(), both given below. */
{
    int x, y, style, a;

    a = win->textAttr;
    window (win->left, win->top, win->right, win->bottom,
            win->textAttr); /* open window */
    if ((style = win->border-1) >= 0) { /* draw border outside */
        gotoxy (win->left-1, win->top-1, 0);
        wrtcha (bord [style][2], a, 0); /* corners: upper left */
        gotoxy (win->right+1, win->top-1, 0);
        wrtcha (bord [style][3], a, 0); /* upper right */
        gotoxy (win->right+1, win->bottom+1, 0);
        wrtcha (bord [style][4], a, 0); /* lower right */
        gotoxy (win->left-1, win->bottom+1, 0);
        wrtcha (bord [style][5], a, 0); /* lower left */
        for (x = win->left; x <= win->right; x++) { /* horizontals */
            gotoxy (x, win->top-1, 0);
            wrtcha (bord [style][0], a, 0);
            gotoxy (x, win->bottom+1, 0);
            wrtcha (bord [style][0], a, 0);
        }
        for (y = win->top; y <= win->bottom; y++) { /* verticals */
            gotoxy (win->left-1, y, 0);
            wrtcha (bord [style][1], a, 0);
            gotoxy (win->right+1, y, 0);
            wrtcha (bord [style][1], a, 0);
        }
        gotoxy (win->left, win->top, 0);
    } /* ----- */
}

void popScroll (POPDESCR *win) /* scroll pop up one line */
{
    winScroll (win->left, win->top, win->right, win->bottom,
                win->textAttr);
    gotoxy (wherex (0), wherey (0) - 1, 0);
} /* ----- */

void popxy (int x, int y, POPDESCR *w) /* gotoxy for popup window */
/* Allows you to express text coordinates relative to upper left */
/* corner of the window in video page 0 */
{
    int row, col;

    row = w->top + y;
    col = w->left + x;
    gotoxy (col, row, 0);
} /* ----- */

void popPuts (int x, int y, char string[], POPDESCR *win) /* write string to */
/* specified window */
{
    int len, ch, p;

    popxy (x, y, win); /* position cursor in window */
    len = strlen (string);
    for (ch = 0; ch < len; ch++) {
        if (string [ch] == '\n') { /* handle newline */
            x = 0;
            ++y;
            popxy (x, y, win);
            if ((y + win->top) > win->bottom) { /* scroll if required */
                popScroll (win);
                --y;
            }
        }
        else {
            if ((x + win->left) > win->right) { /* outside window */
                x = 0; /* so wrap cursor */
            }
            popxy (x, y, win);
            if ((y + win->top) > win->bottom) { /* scroll if required */
                popScroll (win);
                --y;
            }
            wrtcha (string [ch], win->textAttr, 0); /* write next char */
            ++x;
            popxy (x, y, win); /* advance cursor */
        }
    }
} /* ----- */
```

```
char *saveScrn (void) /* saves screen image */
/* Call this routine before popMake(). It saves the current screen */
/* image at a location pointed to by the returned value. You must */
/* pass the same pointer back to restScrn() later in order to */
/* make the pop-up go away. */
{
    int c;
    unsigned srcSeg;
    char *saveArea;

    saveArea = (char *) malloc (4096); /* allocate space */
    if (videomode (&c) == 7)
        srcSeg = 0xB000; /* monochrome buffer */
    else
        srcSeg = 0xB800; /* text graphics buffer */
    movedata (srcSeg, 0, _SS, (unsigned) saveArea, 4096); /* save */
    return (saveArea); /* return pointer */
} /* ----- */

void restScrn (char *saveArea) /* restores screen image */
/* Call this routine when you want the pop up window to go away. */
/* It restores the screen to its appearance before the window. It */
/* does NOT restore the cursor. That is your responsibility. */
/* This routine does not worry about snow on IBM's poorly designed */
/* CGA board. Snow may result when restoring the screen. */
{
    int c;
    unsigned destSeg;

    if (videomode (&c) == 7)
        destSeg = 0xB000; /* monochrome buffer */
    else
        destSeg = 0xB800; /* text graphics buffer */
    movedata (_SS, (unsigned) saveArea, destSeg, 0, 4096); /* restore */
    free (saveArea); /* de-allocate space */
} /* ----- */
```

End Listing One

Listing Two

Listing 2

```
/* MENUBAR.I: Constructs a menu bar per MENUBARSPEC structure */
/* Externally defined are activepage(), videomode(), wrtstra() */
/* Notes: Preserve your cursor position before calling this fn. */
/* It does not save or restore caller's cursor. */
/* The 'sel' pointer points to a solid string of menu */
/* selections in the form "sel1\0sel2\0sel3\0...seln\0" */
/* ----- */
typedef struct {
    int background, foreground; /* colors used in menu bar */
    int nsels; /* number of selections */
    char *sel; /* pointer to static selections (see above) */
} MENUBARSPEC; /* caller sets up as many as needed */
/* does not remember previous cursor position */

void menubar (MENUBARSPEC *spec)
{
    int p, i, ncols, start, interval, page;
    char attr;

    page = activepage (); /* get active page */
    videomode (&ncols); /* get screen width */
    gotoxy (0, 0, page); /* go home */
    attr = chattr (spec->foreground, spec->background); /* attributes */
    for (p = 0; p < (ncols-1); p++) {
        wrtcha (' ', attr, page); /* blank menu bar */
        gotoxy (wherex (page)+1, 0, page); /* advance */
    }
    interval = ncols / spec->nsels; /* spacing between entries */
    start = i = 0;
    for (p = 0; p < spec->nsels; p++) {
        gotoxy (start, 0, page); /* write selections */
        wrtstra (&(spec->sel[i]), attr, page); /* place cursor */
        i += strlen (&(spec->sel[i])) + 1; /* find next string */
        start += interval; /* next position for entry */
    }
}
```

End Listing Two

Listing Three

Listing 3

```
/* MENUDEMO.C: Demonstrates principles of menu bars and pop-down */
/* menus in Turbo C */
/* ----- */
/* INCLUDES */
#include <dos.h>
#include <video.i>
#include <colors.h>
#include <menubar.i>
#include <popwin.i>

/* LOCAL FUNCTION PROTOTYPES */
void popFileMenu (POPDESCR *mfile, char *text);
void popEditMenu (POPDESCR *medit, char *text);
```

(continued on page 108)

Instant Replay

Version III

See us at



COMDEX/Fall '87

November 2-6, 1987

Bally's Las Vegas 8210

Las Vegas, Nevada

SOFTWARE
AHEAD
OF ITS TIME

Instant Replay™ is a unique authoring system for creating:

Demonstrations • Tutorials • Music Presentations • Prototypes • Menus

When active, *Instant Replay™* is a DOS Shell that runs, memorizes, and replays programs. It has the unusual ability to insert prompts, pop-up windows, prototypes, user involvement, music, and branching menus into replays.

Building a Demo or Tutorial is easy. Just run any program with *Instant Replay™* and insert explanation pop-up windows, prompts, user involvement, music and so on. *Instant Replay™* remembers everything; it builds a demo that will re-run the actual program or a screens only prototype version.

Instant Replay™ includes a *Screen Maker* for building pop-up windows, prototype windows, and menu windows. Other useful tools include a *Prototyper*, *Keystroke Editor*, *Music Maker*, *Menu Maker*, *Presentation Text Editor*, *Control Guide*, and *Insertion Guide*.

The screen editor *Screen Genie™* is designed to be memorized by *Instant Replay™*. Creating animations is easy and fun; just run *Screen Genie™* and memorize your activity.

Instant Replay™ for IBM and True Compatibles, requires DOS 2.0 or greater. *Instant Replay™* is not copy protected. **There are no royalties required for distribution of Demos.**

Instant Replay™ at \$149.95 is an exciting new product. Because of the quality of this product, *Nostradamus®* provides a 60-day satisfaction money back guarantee. Call or write, we accept VISA, AmEx, C.O.D., Check or P.O. with orders. Demo diskettes and free product brochure available.

Nostradamus, Inc. 3191 S. Valley Street,
(ste. 252) Salt Lake City, Utah 84109
voice (801) 487-9662

Data/BBS 801-487-9715 1200/2400,n,b,1

Instant Replay™

Features:

- Illustrated manual with index
- Music Maker
- Text Editor
- Online Help
- Screen Painter
- Magic Demo Animator
- Dynamic Menu Maker
- Memorize and Replay actual programs
- Memorize and Replay screens only
- Insert: prompts, pop-ups, prototypes, music, and user involvement into replays
- Make insertions while creating or reviewing
- Generate Vapor Ware from actual programs
- Resident Screen Painter for creating and grabbing windows on the fly
- Prototyper that includes slide shows, menus, and nesting
- Keystroke/time editor, inserter, and merger
- Replay chaining and linking
- Modular demo making facilities
- Fast forward and single step modes
- Self Made Tutorial included
- Timed Keyboard Macros
- Numerous and powerful operator input options for Tutorials
- Text File Presentation facility
- Transparent Windows
- Change Defaults
- Foreground or Background Music
- Canned special sound effects
- Unlimited replay branching
- Compressed screens
- Object oriented programming
- Tracking editor
- Plus much more . . .

"Instant Replay™ is one of those products with the potential to go from unknown to indispensable in your software library." *PC Magazine*

"Incredible . . . We built our entire Comdex Presentation with Instant Replay.™" *Panasonic*

"Instant Replay™ brings new flexibility to prototypes, tutorials, & their eventual implementation." *Electronic Design*

"I highly recommend Instant Replay.™" *Computer Language*

"You need Instant Replay!™" *Washington Post*

Instant Replay™

Instant Assistant™

Screen Genie™

Word Genie™

NoBlink Accelerator™

Assembler Genie™

DOS Assistant™

Programming Libraries

Supports Turbo Pascal 4.0, Turbo C, MS Quick C

HardRunner™

. . . and more

Nostradamus®

Listing Three

(Listing continued, text begins on page 54.)

```

/* STATICS FOR PROGRAM */
char entries[] = {"File\0Edit\0Browse\0Reports\0Exit\0"};
MENUBARSPEC menuspec = {(BLUE), {WHITE}, {5}, {entries}};
char fileMenu[] = {"Open\Save\New file\Abandon"};
char editMenu[] = {"Select record\Remove field\Add field"};
/* -----*/
main ()
{
    static POPDESCR filePop = {{1}, {2}, {8}, {5}, {0}, {2}};
    POPDESCR editPop;
    char *prevScrn;
    int n, oldx, oldy, cols;

    /* SET UP POP-DOWN MENU FOR FILE SELECTION */
    filePop.textAttr = chattr (YELLOW, BLUE);

    /* SET UP POP-DOWN MENU FOR EDIT SELECTION */
    editPop = filePop;
    editPop.left = 80 / menuspec.nsels + 1;
    editPop.right = editPop.left + 12;
    editPop.bottom = editPop.top + 2;

    /* NOW SHOW THE TWO MENUS IN SUCCESSION */
    setmode (3);
    cls ();
    menubar (&menuspec); /* write menu */
    printf ("\nActive page is %d", activepage()); /* show video info */
    printf ("\nVideo mode is %d", videomode (&cols));
    printf ("\nNumber of columns on screen is %d", cols);
    oldx = wherex (activepage());
    oldy = wherey (activepage());
    printf ("\nCursor is at x = %d, y = %d", oldx, oldy);
    gotoxy (oldx, oldy, activepage());
    getch (); /* wait for keypress */
    cursoff ();
    popFileMenu (&filePop, fileMenu); /* pop down file menu */
    popEditMenu (&editPop, editMenu);
    setcursor (0, cursend()); /* make a block cursor */
    gotoxy (oldx, oldy, activepage());
    puts ("\nPress any key to end demo...");
    getch ();
    setcursor (cursend()-1, cursend()); /* restore underline cursor */
    cls ();
} /* -----*/
void popFileMenu (POPDESCR *mfile, char *text)
{
    char *prevScrn;
    prevScrn = saveScrn ();

```

```

    popMake (mfile);
    popPuts (0, 0, text, mfile);
    getch ();
    restScrn (prevScrn);
} /* -----*/
void popEditMenu (POPDESCR *medit, char *text)
{
    char *prevScrn;

    prevScrn = saveScrn ();
    popMake (medit);
    popPuts (0, 0, text, medit);
    getch ();
    restScrn (prevScrn);
} /* -----*/

```

End Listings

Dr. Dobb's Back Issues

January 1987 #123 Volume XII, Issue 1
Annual 68k Issue—68K Mini Forth, OS-9 Operating System—MAC and Amiga Interface Programming

February 1987 #124 Volume XII, Issue 2
Editors and Assemblers

May 1987 #127 Volume XII, Issue 5
Notes on Computer Music—Scientific Programming—Command Processors

June 1987 #128 Volume XII, Issue 6
Handling Large Priority Queues—TSR Serial Drivers—UNIX Shell Scripts

July 1987 #129 Volume XII, Issue 7
386 Development Tools—Optimizing 8088 Code—Curses for MS-DOS

August 1987 #130 Volume XII, Issue 8
Unveiling ANSI C—New Tools for C—Ray Duncan on DOS 3.3—AI: Programing in Loops

September 1987 #131 Volume XII, Issue 9
Quest for Algorithms—Writing MS-DOS Device Drivers

October 1987 #132 Volume XII, Issue 10
Stretching AppleTalk—Focus on Forth: Unifying Dialects, Faster Forth, A New Forth Column—Quick C & Turbo C

November 1987 #133 Volume XII, Issue 11
Special Graphics Issue—Tools for: 3-D Mapping, Screen Management, Turbo C Graphics

December 1987 #134 Volume XII, Issue 12
Making Sense of Operating Systems—Dynamic Linking in OS/2, ROMing C Code, Turbo C Graphics, Languages: C, Assembler, Forth

Other issues are also available. Please inquire.

TO ORDER: Return this coupon with your payment to:
M&T Books, 501 Galveston Drive, Redwood City, CA 94063. Or, call TOLL-FREE 800-533-4372 (In CA 800-356-2002)

Price: 1 issue \$5.00; 2-5 issues \$4.50 each; 6 or more \$4.00 each. There is a \$10 minimum for charge orders.

Name _____

Address _____

City _____ State _____ Zip _____

Please send the issues circled:

96 97 104 108 109 113 114 115 118 119 120
121 122 123 124 127 128 129 130 131 132
133 134

Subtotal _____

Outside U.S. add \$.50 per issue _____

TOTAL _____

_____ Check enclosed. Make payable to M&T Books
Charge my _____ VISA _____ M/C _____ AmerExp

Card # _____ Exp.Date _____

3134

New Horizons...for dBASE Users.

Resident...Compatible...Powerful — FrontRunner offers all this and more.

- **CREATE MEMORY RESIDENT dBASE PROGRAMS.** FrontRunner is the first memory-resident applications development tool which uses the popular dBASE programming language.
- **BUILT-IN COMPILER.** FrontRunner applications can be compiled into binary files for protected distribution and unparalleled speed.
- **dBASE III+ LANGUAGE AND FILE COMPATIBILITY.** Standard commands, functions, and syntax are provided, along with complete data and index file compatibility, allowing you to use FrontRunner immediately.
- **UNIQUE POWERKEY FEATURE.** Bind commands or entire programs to a single PowerKey for rapid execution from within other applications.
- **PASTE COMMAND.** This powerful command allows extraction of data into your application, using standard filtering expressions.

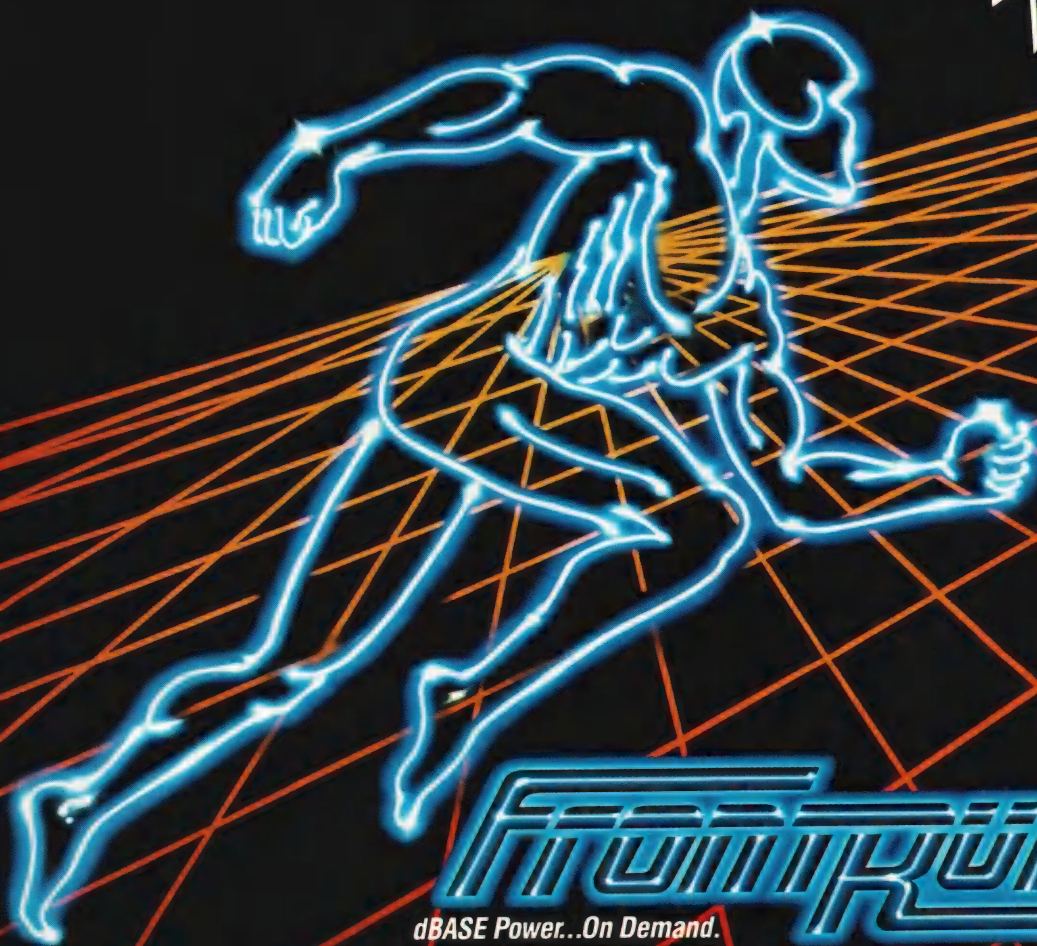
dBASE and dBASE III+ are registered trademarks of Ashton-Tate, Inc.

The MS-DOS version of FrontRunner is \$295.00. Orders received before December 1, 1987 include an unlimited run-time license, allowing you to distribute royalty-free applications throughout the universe. (After this date the unlimited run-time package will be sold separately.) FrontRunner is not copy-protected, and comes with a 30-day money back guarantee. To order, call the number below and get a running start toward more exciting and productive applications development!

APEX

Apex Software Corporation
4516 Henry Street
Pittsburgh, PA 15213
(412) 681-4343

Now
Available
From
Ashton-Tate, Inc.



dBASE Power...On Demand.

CIRCLE 185 ON READER SERVICE CARD

Listing One (Text begins on page 126.)

Listing 1 -- kernel.h, Printed 9/11/1987

```

1 | #ifndef NULL
2 | #include <stdio.h>
3 | #endif
4 | #include <tools/pq.h>
5 |
6 |
7 | #define TE_NOERR          0      /* Error codes */
8 | #define TE_TOOMANY       -1     /* No error */
9 | #define TE_NOMEM         -2     /* Maximum number of tasks (32) already exists */
10 | #define TE_BADARG        -3     /* Insufficient memory available */
11 | #define TE_TIMEOUT       -4     /* Illegal Argument */
12 | #define TE_QFULL        -5     /* Timeout */
13 | #define TE_NOTASKS       -6     /* Queue is full */
14 | #define TE_INTERNAL      -7     /* No tasks to send message */
15 | #define TE_DEADLOCK      -8     /* Internal error */
16 | #define TE_STACK         -9     /* Delete would have caused a deadlock. */
17 | #define TE_KILL          -10    /* Stack overflow */
18 | /* Ctrl-Break encountered */
19 |
20 | #define TS_NORMAL         0      /* Must be 0 */
21 | #define TS_WAIT           1
22 | #define TS_TIMEOUT        2
23 |
24 | #define T_MAXTASK         32    /* Max. number of tasks that can be active */
25 | #define TQ_SIG            0xa5a5 /* Signature used for queues to test validity */
26 |
27 | /* PRIORITY(a,b) evaluates to a neagive number if task a is lower priority
28 | * than task b, to 0 if they're equal, to a positive number
29 | * if task a is higher priority than task b. If priorities
30 | * are the same, the timestamps are compared and the routine
31 | * with the smaller (older) time stamp is assumed to be the
32 | * higher priority.
33 | */
34 | #define T_PRIORITY(a,b) ( ((a)->priority != (b)->priority) \
35 |                          ? (a)->priority - (b)->priority \
36 |                          : (b)->timestamp - (a)->timestamp )
37 |
38 | /* -----
39 | * Task Control Block. Do not change the register-save area
40 | * (ax, bx, cx ... ) without also changing the code in swap.asm.
41 | * Don't change anything without changing the offset to the stack
42 | * base in chkstk.asm.
43 | *
44 | * I'm assuming the small model here. That is, I'm assuming that
45 | * the only segment register that can change is the extra segment
46 | * and that the stack and data segments always have the same value.
47 | *
48 | * Be sure to block() if you're going to modify the CS, DS, or SS
49 | * registers.
50 | *
51 | * A context swap is done by pushing the registers in the following
52 | * order:
53 | *      flags,cs,ip,ax,bx,cx,dx,si,di,bp,ds,es
54 | *
55 | * Then, the current stack pointer is saved in the TCB. Context is
56 | * restored by popping es,ds,bp,di,si,dx,cx,bx, and ax, and then
57 | * restoring the flags, cs, and ip with an iret instruction.
58 | */
59 |
60 | typedef struct tcb
61 | {
62 |     void **sp; /* Must be first and must be 16 bits */
63 |     unsigned ss; /* Must be second & must be 16 bits */
64 |
65 |     unsigned priority; /* priority 0=lowest, 65,535=highest */
66 |     unsigned long timestamp; /* Clock tick when task was preempted. */
67 |
68 |     unsigned wait; /* Counting semaphore used by tasks that
69 |                    * are waiting at a queue. Set to initial
70 |                    * timeout value and decremented on each
71 |                    * clock tick. Task is put back into
72 |                    * the active list if semaphore gets to
73 |                    * 0. If wait < 0, task will not time out.
74 |                    */
75 |
76 |     struct tcb *next; /* Pointer to next task waiting at queue. */
77 |
78 |     int status; /* TS_NORMAL Not suspended by wait.
79 |                * TS_WAIT Suspended by wait
80 |                */
81 |
82 |     void *msg; /* Dequeued message if task was waiting
83 |                * for a message. NULL if task timed out.
84 |                */
85 |
86 |     /* The following are handy for debugging */
87 |     /* but aren't used for anything else */
88 |     char *tag; /* Identifying string of some sort */
89 |     void **initial_sp; /* Initial stack pointer */
90 |
91 |     void *stack[1]; /* First cell of stack. Must be last
92 |                    * thing in the structure. Must be declared
93 |                    * as pointer-sized for t_create().
94 |                    */
95 | }
96 | TCB;
97 |
98 | typedef struct t_queue
99 | {
100 |     int signature; /* Signature */
101 |     struct t_queue *next; /* Next queue in chain. */
102 |     TCB *task_h; /* Head (start) of task list. */
103 |     TCB *task_t; /* Tail (end) of task list. */
104 |     int q_size; /* Maximum number of elements */
105 |     int numele; /* # of elements currently in queue */
106 |     void **headp; /* Head pointer */

```

(continued on page 113)

Oasys presents Microsoft[®] *cross* C for VAX, Sun, Apollo . . .



OASYS is proud to announce the immediate availability of the OASYS/Microsoft Cross C Development System. Microsoft C, MASM (Assembler), and LINK (Linker) now run on DEC VAX (VMS and Ultrix), Sun and Apollo systems.

Those accustomed to using these superior Microsoft tools on a PC can now build MS-DOS applications on a VAX or workstation. OASYS guarantees that the unsurpassed speed, compactness, and flexibility of Microsoft C have been preserved. The OASYS/Microsoft Cross C Development System offers identical functionality to Microsoft C -- no short-cuts, no alterations -- repackaged to meet today's demands for high performance/low cost development on non-MS-DOS systems.

With the OASYS/Microsoft Cross C Development System you can maintain, or even extend, applications originally created on a PC. Software development teams can now build large, complex MS-DOS (soon OS/2) applications on powerful centralized VAXs or networked workstations.

Regardless of where you choose to do development, OASYS provides the best tools, on the widest variety of hosts, with comprehensive support. Our exclusive relationship with Microsoft, the world's leading supplier of MS-DOS software products, is evidence of our commitment to provide evolving PC tools to OASYS customers.

Prices start at \$1,000. New ports are underway. Call today for more information. OEM and end-user inquiries are encouraged.

Oasys

Microsoft[®]

CIRCLE 186 ON READER SERVICE CARD

230 Second Avenue, Waltham, MA 02154 (617) 890-7889

MS-DOS, Microsoft and the Microsoft logo are registered trademarks of Microsoft Corp. Apollo is a trademark of Apollo Computer Inc. Trademarks are also acknowledged to DEC, Sun Microsystems, Inc., XEL, Inc.

ALL GAIN, NO PAIN

Blow away the 640K barrier

Gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and Assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode.

OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors, with 32-bit capability *today* on 386s that yields twice the performance of 16-bit mode. OS/286 and OS/386 have quickly become the preferred solution for developers of high performance, memory-intensive applications, including CADKEY, CASE, VIEWlogic, and Gold Hill, and premier language developers Lahey and Metaware.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?)

If your applications are running out of memory or need more speed, don't wait for the "solution" that means abandoning your investment in DOS. Enhance them now with OS/286 and OS/386 — *products not promises.*

**Make
big programs run faster
in protected mode**



OS/286™ & OS/386™ Benefits:

- Gain multi-megabytes of directly addressable memory (15-Mb-286, 4Gb-386)
- Increase performance by eliminating overlays and EMS
- Convert in days, not months
- Continue to work with a DOS interface and use existing TSRs, device drivers, graphic routines, etc.
- Stay compatible with the widest array of systems

A.I. Architects Software Developers Kit \$495

includes full support for:

- MetaWare High C (16 & 32 bit)
- Professional Pascal (16 & 32 bit)
- Lahey F77L FORTRAN
- Microsoft C & Fortran 4.0,
- MASM, MS-Link
- Phoenix PLINK86
- Halo & GSS Graphics
- Pharlap 386: ASM/LINK more to come

Run time licenses for OS/286 and OS/386 are available at nominal cost.

The HummingBoard™

turns any XT or AT into the fastest 386 system available. The dual processor architecture boosts performance significantly over comparable single processor systems or accelerator boards. Available with 2 to 24Mb RAM, 16 or 20Mhz speed, and 387 floating point coprocessor.



**A.I.
Architects, Inc.**

One Kendall Square, Building 400
Cambridge, Massachusetts 02139
(617) 577-8052

OS/286, OS/386 and HummingBoard are trademarks of A.I. Architects, Inc., High C and Professional Pascal are trademarks of Metaware, Inc., F77L FORTRAN is a trademark of Lahey Computer Systems, Inc., Microsoft and MS-DOS are trademarks of Microsoft Corp.

CIRCLE 187 ON READER SERVICE CARD

C CHEST

Listing One (Listing continued, text begins on page 126.)

```

107| void          **tailp;    /* Tail pointer          */
108| void          *queue[1];  /* First cell of actual queue.
109|                      * Must be at the bottom of the
110|                      * structure.
111|                      */
112| }
113| T_QUEUE;
114|
115| /*-----
116| * Global variables. Actually declared in globals.c. I'm assuming
117| * the default initialization to 0 here. These may be used by
118| * your programs (T_clock and T_numtasks are useful) but should
119| * never be modified by them. It's safest to block while
120| * accessing them.
121| */
122|
123| #ifdef ALLOC
124| #   define CLASS
125| #   define I(x) x
126| #else
127| #   define CLASS extern
128| #   define I(x)
129| #endif
130|
131| CLASS PQ      *T_tasks I(=0); /* Priority queue of tasks that are waiting */
132| /* for service. See /src/tools/pq.c for priority */
133| /* queue routines and definition of PQ. */
134|
135| CLASS TCB     *T_active I(=0); /* Pointer to currently active task. NULL if
136| * multitasking is off or if no tasks are active
137| * (this latter is a deadlock).
138| */
139|
140| CLASS unsigned long T_clock I(=0);
141|
142| /* Incremented on each system clock tick. If you
143| * assume the default 18.2 ticks/second,
144| * the clock will roll over after about
145| * 65552 hours (about 7.47 years):
146| *
147| * ((0xffffffff/18.2) /60) /60 == 65552
148| * 65552 / 24 /365.35 == 7.47798
149| *
150| * Of course, this number will scale with faster
151| * tick rates but the resolution should be ok for
152| * all reasonable tick rates.
153| */
154|
155| CLASS T_QUEUE *T_queues I(=0); /* Pointer to head of linked list of queues. */
156| CLASS int      T_numtasks I(=0); /* Total # of tasks that have been created. */
157|
158| /*-----
159| * Function prototypes. You should never call any of the _t_xxxx
160| * functions directly.
161| */
162|
163| extern T_QUEUE *t_makequeue (int size, void *msg, void *q, int timeout);
164| extern int t_send (T_QUEUE *q, void *msg, int timeout);
165| extern void *t_wait (T_QUEUE *q, int timeout);
166| extern int t_yield (void);
167| extern int t_perror (char *str, int errcode);
168| extern int t_start (int factor);
169| extern TCB *t_create (int (*subr)(), char* tag, unsigned pri, int stk_size,...);
170|
171| extern int t_chg_priority (TCB *tp, int new_priority);
172| extern int t_delete (TCB *task);
173| extern int t_print (TCB *task);
174| extern void t_stop (int exit_code);
175| extern int t_second (void);
176| extern void t_swap (TCB *old, TCB *new);
177| extern void t_install (TCB *new);
178| extern void t_shazam (void);

```

End Listing One

Listing Two

Listing 2 -- schedule.asm, Printed 9/11/1987

```

1| PAGE 56,132
2| TITLE SPEEDUP.ASM: System-clock-modification routines
3| ;-----
4|
5| DEBUG equ 1 ; Set to 1 to make internal symbols public for
6| ; debugging.
7|
8| DOSPEED equ 1 ; Set to 0 to disable everything except
9| ; global-variable initialization in speedup.
10|
11| ;-----
12| ; Public Subroutines are:
13| ;
14| ; t_cli()
15| ; Disable Interrupts
16| ; t_sti()
17| ; Enable Interrupts
18| ;
19| ;-----
20| ; t_speedup( factor )
21| ; Int factor;
22| ;

```

(continued on next page)

MINIMIZE TURBO PASCAL DEBUG TIME

Tmark allows Turbo to continue compiling after an error is found without returning to line one!

Picture this:

You have just added a new routine at line 2,000 to an existing program. The routine contains 5 simple errors. You are going to have to compile those first 2,000 lines 5 times in order to find all the errors. That's 10,000 line of code!

But now you compile that program with Tmark installed. You will still have to compile the first 2,000 lines in order to find the first error. However, when you fix that error Turbo will continue compiling where you left off, not back at line one. You will be able to skip from error to error with no recompiling!

Tmark creates image files on disk to save and restore Turbo's state during a compilation. Saves are made automatically before compiler errors or at lines designated with a {tmark} comment. A window pops up to let you select which image to use to resume the compilation.

Tmark dramatically reduces debug time. Once you try it you will never want to give it up! \$80 + \$2 s/h, Visa/MC

TANGENT DESIGNS

PO Box 896, Lake Forest, IL 60045

(800) 356-2750, (312) 295-0030

CIRCLE 188 ON READER SERVICE CARD

PLOT TEXT ON ANY GRAPHICS SCREEN!!

YES, we said ANY Graphics Screen, even VGA! FINALLY! Xgraf is a super set of smart low level assembly graphic routines that you call directly from Compiled BASIC. Xgraf replaces BASIC's confusing graphics statements with consistent, full featured calls specifically designed for the BASIC programmer.

FINALLY! Xgraf is only \$99.00 + \$4.00 S&H

We specialize in libraries and tools for Compiled BASIC. Our catalog features the FINALLY! Family of Products and other top flight tools.

Call: 1 800 423-3400

(9:00 AM to 8:00 PM EST)

PA & AK call (412) 782-0384



KOMPUTERWERK

851 Parkview Blvd.
Pittsburgh, PA 15215

CIRCLE 189 ON READER SERVICE CARD

In conjunction with Glockenspiel of Dublin

C++ TRAINING

*Programming Workshop

December 14-18

January 11-15

February 22-26

*Overview for Developers

December 9

January 20

February 10

On site courses available.

For more information and reservation call:

Semaphore, Inc.

16 Haverhill Street, Andover, MA 01810

(617) 474-0040

CIRCLE 190 ON READER SERVICE CARD

Listing Two

(Listing continued, text begins on page 126.)

```

23;      Speed up the system clock by the indicated factor
24;      (1, 2, 3, 4, etc.). Call the scheduler
25;      on every timer interrupt and call the default clock
26;      routine as well every "factor" ticks. For example,
27;      speedup(2) speeds up the system clock by
28;      a factor of two; the normal interrupt-service
29;      routine that's used by DOS will be called every-
30;      other tick. A speedup factor of 1 or 0 doesn't modify
31;      the clock rate.
32;
33; _t_slowdown()
34;
35;      Restore the clock to the normal speed and disconnect
36;      the routine installed with a previous speedup() call
37;      (if one is installed).
38;
39;
40; t_block()
41; t_release()
42;
43;      Disable the scheduler but not the normal clock interrupt.
44;      The default system interrupt-service routine is processed
45;      in the normal way, on every Nth clock tick. Use these
46;      routines carefully. If they're used in a tight loop, it's
47;      possible that ALL timer interrupts will be ignored, even
48;      if the processor is released for a while inside the loop.
49;      In this situation sti() and cli() are better.
50;
51;
52; long numint();      Number of unblocked interrupts.
53; long numblk();      Number of blocked interrupts.
54;
55;
56; TEXT SEGMENT BYTE PUBLIC 'CODE'
57; TEXT ENDS
58; DATA SEGMENT WORD PUBLIC 'DATA'
59; DATA ENDS
60; CONST SEGMENT WORD PUBLIC 'CONST'
61; CONST ENDS
62; BSS SEGMENT WORD PUBLIC 'BSS'
63; BSS ENDS
64; DGROUP GROUP CONST, BSS, DATA
65; ASSUME CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP
66;
67; EXTRN _chkstk:NEAR
68; EXTRN _t_reschedule:NEAR
69; EXTRN _t_active:WORD
70;
71;
72;
73; TIMR_CTRL = 43H      ; address of timer control port
74; TIMR_0_DATA = 40H    ; address of counter 0 data port
75; TIMR_0_LOAD = 36H    ; control word for timer
76;
77; STKSIZE = 256        ; Number of bytes on interrupt-
78;                      ; service-routine stack
79;
80;
81;
82; TEXT SEGMENT
83;
84;
85;      Misc. variables. Note that I'm putting all these in the
86;      code (TEXT) segment so that I can find them when an
87;      interrupt comes along. The PUBLIC statements are just for
88;      debugging.
89;
90; old_int equ $
91; old_off dw ?      ; Offset of old timer interrupt routine.
92; old_seg dw ?      ; Segment address of same.
93;
94; tick_reset dw ?
95; numticks dw ?      ; Initialized to tick_reset, decre-
96;                      ; mented on each timer interrupt,
97;                      ; reset to the speedup factor (and the
98;                      ; old service routine is called) when
99;                      ; it reaches zero.
100;
101; stack db STKSIZE dup (0) ; Local stack for service routine
102;                      ; 30 bytes are used by real service
103;                      ; routine, the rest is available
104;                      ; for the user service routine.
105; stack_end dw ?
106;
107; old_ds dw ?
108; old_sp dw ?
109; old_ss dw ?
110; old_ax dw ?
111; old_ip dw ?
112; old_cs dw ?
113; old_fi dw ?
114;
115; numint dd 0      ; total number of interrupts
116; numblk dd 0      ; number of times user routine blocked
117;
118; blocked db 0      ; don't execute user routine if true
119;
120; IF DEBUG
121; PUBLIC old_int, old_off, old_seg, old_ds,
122; PUBLIC tick_reset, numticks, stack, stack_end, old_sp,
123; PUBLIC old_ss, old_ax, old_ip, old_cs, old_fi
124; PUBLIC blocked, serv, numint, numblk
125; ENDIF
126;
127;
128; ; statistics stuff.
129;
130; PUBLIC _t_numint, _t_numblk
131;
132; _t_numint PROC NEAR
133; mov ax,WORD PTR cs:numint
134; mov dx,WORD PTR cs:numint+2

```

```

135; ret
136; _t_numint ENDP
137;
138; _t_numblk PROC NEAR
139; mov ax,WORD PTR cs:numblk
140; mov dx,WORD PTR cs:numblk+2
141; ret
142; _t_numblk ENDP
143;
144;
145; ; t_cli(); t_sti(); Disable and enable interrupts.
146;
147; PUBLIC _t_cli, _t_sti
148;
149; _t_cli PROC NEAR
150; cli
151; ret
152; _t_cli ENDP
153;
154; _t_sti PROC NEAR
155; sti
156; ret
157; _t_sti ENDP
158;
159;
160;
161; t_block(); Disable and enable user interrupt service
162; t_release(); routine but not the real interrupt service
163; routine (or the interrupt itself).
164;
165; PUBLIC _t_block, _t_release
166;
167; _t_block PROC NEAR
168; mov byte ptr cs:blocked,1
169; ret
170; _t_block ENDP
171;
172; _t_release PROC NEAR
173; mov byte ptr cs:blocked,0
174; ret
175; _t_release ENDP
176;
177;
178; ; t_speedup( factor )
179; ; Int factor;
180;
181; factor = [bp+4]
182; routine = [bp+6]
183;
184; PUBLIC _t_speedup
185;
186; _t_speedup PROC NEAR
187; push bp
188; mov bp,sp
189; xor ax,ax
190; call _chkstk
191; mov _TEXT:old_ds,dx ; remember current DS.
192; mov ax,[bp+4] ; AX = factor
193; mov _TEXT:tick_reset,ax ; tick_reset = factor;
194; mov _TEXT:numticks,ax ; numticks = factor;
195;
196; if DOSPEED
197;
198; mov ax,[bp+4] ; if( factor && factor != 1)
199; cmp ax,01H ; {
200; je noload ;
201; cmp ax,00H ;
202; je noload ;
203;
204; mov al,TIMR_0_LOAD ; Set up timer for load
205; out TIMR_CTRL,al ;
206; mov ax,00000H ;
207; mov dx,00001H ; Number of ticks
208; mov bx,[bp+4] ; BX = factor.
209; div bx ; AX = number of ticks
210; cli ;
211; out TIMR_0_DATA,al ; Send new count to timer
212; mov al,ah ;
213; out TIMR_0_DATA,al ;
214; sti ;
215; noload:
216;
217; mov ah,35H ; Get the old vector
218; mov al,00H ;
219; int 21H ;
220; mov _TEXT:old_off,bx ;
221; mov _TEXT:old_seg,es ;
222;
223; ; set up the new vector
224; mov ah,25H ;
225; mov al,00H ;
226; mov dx,OFFSET _TEXT:serv ;
227; push ds ;
228; push cs ;
229; pop ds ;
230; int 21H ;
231; pop ds ;
232; endif
233; mov sp,bp
234; pop bp
235; ret
236;
237; _t_speedup ENDP
238;
239;
240;
241; PUBLIC _t_slowdown
242;
243; _t_slowdown PROC NEAR
244;
245; push bp
246; mov bp,sp
247; xor ax,ax
248; call _chkstk

```



```

249|
250|     mov     ax,_TEXT:old_off ; See if the interrupts have
251|     or      ax,ax           ; changed.
252|     jz      no_int         ; No, don't fix them then
253|
254|                                     ; restore old timer interrupt
255|     push    ds
256|     mov     ah,25H
257|     mov     al,08H
258|     mov     ds,_TEXT:old_seg ;
259|     mov     dx,_TEXT:old_off ;
260|     int     21H
261|     pop     ds
262|
263| no_int:  mov     al,TIMR_0_LOAD ; Restore default system
264|          out     TIMR_CTRL,al   ; clock tick rate
265|          mov     al,0
266|          out     TIMR_0_DATA,al
267|          out     TIMR_0_DATA,al
268|
269|          mov     sp,bp
270|          pop     bp
271|          ret
272|
273| _t_slowdown ENDP
274|
275| -----
276| ; Actual interrupt service routine.
277| ; Note that the flags, cs, and ip are pushed on entry (because
278| ; if the interrupt)
279|
280| serv PROC NEAR
281|
282|     push    ax
283|
284|     mov     al,byte ptr cs:blocked ; If( servicing blocked)
285|     or      al,al
286|     jz      serv1
287|
288|     add     WORD PTR cs:numblk,1 ; ++numblk;
289|     adc     WORD PTR cs:numblk+2,0 ;
290|     jmp     servexit
291|
292| serv1:
293|     add     WORD PTR cs:numint,1 ; {
294|     adc     WORD PTR cs:numint+2,0 ; ++numint;
295|     push    bx
296|     push    cx
297|     push    dx
298|     push    si
299|     push    di
300|     push    bp
301|     push    ds
302|     push    es
303|
304|     mov     ds,_TEXT:old_ds ; Restore Data segment
305|     mov     bx,T_active ; BX = T_active
306|     mov     [bx],sp ; T_active->sp = SP
307|     mov     [bx+2],ss ; T_active->ss = SS
308|
309|     push    cs ; Set up local stack
310|     pop     ss
311|     mov     sp,offset _TEXT:stack_end ;
312|
313|     call    _t_reschedule ; in task.c
314|
315|     mov     bx,T_active ; BX = new T_active,
316|     mov     ss,[bx+2] ; will be the same as
317|     mov     sp,[bx] ; the old T_active if
318|     pop     es ; no change is reqd.
319|     pop     ds
320|     pop     bp
321|     pop     di
322|     pop     si
323|     pop     dx
324|     pop     cx
325|     pop     bx
326|
327| servexit:
328|     dec     _TEXT:numticks ; if(--numticks > 0)
329|     jle     serv3 ; {
330|     mov     al,20h ; send EOI
331|     out     20h,al ;
332|     pop     ax
333|     iret ; }
334|
335| serv3:
336|     mov     ax,_TEXT:tick_reset ; numticks = tick_reset;
337|     mov     _TEXT:numticks,ax ;
338|     mov     ax,ax ;
339|     jmp     dword ptr _TEXT:old_int ; jmp to old vector
340|
341| serv ENDP
342|
343|
344| _TEXT ENDS
345| END

```

End Listing Two

(Listing Three begins on page 116.)

Introductory Offer \$99.95

The added plus you need for developing sophisticated computer graphics, CAD, and programs that use computational geometry.

You save time and money with its flexibility.

TurboGeometry Library


An excellent addition to Borland's Turbo Graphix Toolbox.

Over 150 ready to use routines, such as Equations of Lines, Circles, Arcs & Planes • Intersection of Lines, Circles, Planes • Curves • 2&3 Dimensional Transforms • Polygons • Hidden Lines • Volumes • Areas • Perspectives • Polygon Clipping and more....

Manual, full source code and sample programs. All for the introductory price of \$99.95US. Add \$5.00 for SH. Tex Res add 8% ST. 30 day money back guarantee. VISA, MC, Check MO accepted. Needs Turbo Pascal 2.0+, IBM PC(Comp), Zenith Z100, MS/PC DOS 2.0+.

DISK SOFTWARE, INC., 2116 E Arapaho, # 487 Richardson, Texas 75081 (214) 423-7288

CIRCLE 191 ON READER SERVICE CARD



MPULSE COMPUTERS
THE NEXT GENERATION IN
UNIX* PROGRAM DEVELOPMENT

MPulse Model 21 (\$14,995 list)

- * Support for 64 users
- * 1 Gigabyte storage capacity
- * 32 bit processing power
- * Multiprocessor architecture
- * OS derived from AT&T UNIX System V

MPulse Model 20 (\$9,995 list)

- * Support for 32 users
- * 0.5 Gigabyte storage capacity
- * 32 bit processing power
- * Multiprocessor architecture
- * OS derived from AT&T UNIX System V

MPulse is the first computer system ever to offer minicomputer performance for under \$10,000. Call today to find out more about MPulse computers, and ask about our software developer discounts. Call (214) 340-5172

Logic Process Corporation 10355 Brockwood Road Dallas TX 75238

CIRCLE 192 ON READER SERVICE CARD

UNIX is a trademark of AT&T

Listing Three

(Listing continued, text begins on page 126.)

Listing 3 -- swap.asm, Printed 9/11/1987

```

1 ; SWAP.ASM      Routine to do context swaps. Everything in
2 ;              this file is VERY compiler dependant and
3 ;              VERY nonportable.
4 ;
5 ;              TITLE      swap.c
6 ;              NAME       swap
7 ;
8 ; TEXT SEGMENT WORD PUBLIC 'CODE'
9 ; TEXT ENDS
10 ; DATA SEGMENT WORD PUBLIC 'DATA'
11 ; DATA ENDS
12 ; CONST SEGMENT WORD PUBLIC 'CONST'
13 ; CONST ENDS
14 ; BSS SEGMENT WORD PUBLIC 'BSS'
15 ;
16 op      db      0      ; Used by rst_chkstk and chg_chkstk
17 segm    dw      0      ; (below).
18 off     dw      0
19
20 save_bx dw      0      ; used by __t_swap_in
21
22 BSS ENDS
23 DGROUP GROUP CONST, BSS, DATA
24 ASSUME CS: _TEXT, DS: DGROUP, SS: DGROUP
25
26 DATA SEGMENT
27 _DATA ENDS
28
29 IF 1
30 PUBLIC stack_err, mychkstk, rst_chkstk, chg_chkstk
31 PUBLIC op, segm, off
32 ENDIF
33
34 ;-----
35 ; TEXT SEGMENT
36 ;
37 ; ASSUME CS: _TEXT
38 ;
39 PUBLIC __t_swap_in ; Swaps two tasks
40 PUBLIC __t_install ; Install a task when none active
41 PUBLIC __t_shazam ; Starts multitasking
42 PUBLIC __t_stop ; Stops multitasking.
43 PUBLIC __t_suspend ; Temporarily suspend stack checking.
44 PUBLIC __t_rst_chkstk ; Restore it again.
45
46 EXTRN __chkstk:NEAR ; In standard library
47 EXTRN __free:NEAR ; In standard library
48 EXTRN __t_slowdown:NEAR ; In schedule.asm
49 EXTRN __t_block:NEAR ; "
50 EXTRN __t_release:NEAR ; "
51 EXTRN __t_isrerr:NEAR ; In task.c
52 EXTRN __T_active:WORD ; Declared in kernel.h. Pointer
53 ; to currently active task
54
55 ;-----
56
57 save_sp dw 0
58 save_ss dw 0
59 chk_on dw 0 ; No stack probes while nonzero
60
61 ;-----
62
63 __t_shazam PROC NEAR
64
65 ; Start the ball rolling. Save the current context.
66 ; then start up the first task. On entry,
67 ; T_active must point at the first task to activate.
68 ; Stack on entry (from top to bottom) is:
69 ;
70 ; return address from __t_shazam call
71 ; old bp saved by __t_start
72 ; t_start's return address
73 ; speedup_factor passed to __t_start
74 ;
75
76 add sp,2 ; Discard return addr to __t_shazam
77 ; Uncovering bp from main that
78 ; was saved by __t_start()
79
80 push si
81 push di
82 push ds ; push this last
83
84 mov WORD PTR cs:save_sp, sp
85 mov WORD PTR cs:save_ss, ss
86
87 call chg_chkstk
88 mov bx, WORD PTR __T_active
89 jmp shazam
90
91 __t_shazam ENDP
92
93 ;-----
94
95 __t_stop PROC NEAR
96
97 ; __t_stop( errcode )
98 ;
99 ; This routine deletes the current task, causes
100 ; multitasking to be turned off, and passes control
101 ; back to the routine that called __t_start()
102 ; (immediately following the __t_start call).
103 ; Errcode is passed back to the calling routine as the
104 ; return value of __t_start().
105 ;
106 ; It can be called directly by a running task; it's
107 ; called automatically when the last task is deleted,
108 ; or when the only running task deletes itself.
109
110 cli ; Just to make sure
111
112 add sp,2 ; Discard return address
113 pop ax ; return value = errcode
114
115 mov ss, WORD PTR cs:save_ss ; Restore initial stack...
116 mov sp, WORD PTR cs:save_sp ; ...and data segment.
117 pop ds
118
119 push ax
120 call rst_chkstk ; Put back original __chkstk
121 call __t_slowdown ; Disable weird timer int.
122
123 pop ax ; get back return value
124 push ax
125
126 or ax,ax ; if( errcode == NOERR )
127 jnz t_stopl ; {
128 push __T_active ; free( T_active );
129 call __free ;
130 add sp,2 ; }
131
132 t_stopl:
133 pop ax ; return( errcode )
134
135 pop di ; restore si and di saved by
136 pop si ; by __t_shazam,
137 pop bp ; and bp saved by __t_start.
138 ret
139 __t_stop ENDP
140
141 ;-----
142
143 __t_install PROC NEAR
144
145 ; t_install( new )
146 ; TCB *new;
147 ;
148 ; Delete the current task and replace it with the
149 ; new one. This routine saves some space (and execution
150 ; time) by jumping into the middle of swap() to install
151 ; the new task. The scheduler must be blocked when
152 ; this routine is called. This routine does not
153 ; return.
154
155 add sp,2 ; discard return address
156 pop bx ; bx = new
157
158 mov ss, WORD PTR [bx+2] ; ss:sp = new task's stack;
159 mov sp, WORD PTR [bx]
160
161 push bx
162 push __T_active ; free( T_active )
163 call __free
164 add sp,2 ; Discard arg to free()
165 pop bx ; get back bx.
166
167 jmp shazam
168
169 __t_install ENDP
170
171 ;-----
172
173 __t_swap_in PROC NEAR
174
175 ; __t_swap_in( new )
176 ; TCB *new;
177 ;
178 ; Do a context swap. Replace T_active with new. This
179 ; routine returns only when the original context is
180 ; restored. Swapping MUST be blocked when this subroutine
181 ; is called. Release() is called once the new context
182 ; is installed and T_active is modified to point at the
183 ; new task.
184
185 cli
186 mov WORD PTR cs:save_bx, bx
187 pop bx ; bx = return address
188 pushf ; Save current context
189 push cs ; (Push return address as new ip)
190 push ax
191 push save_bx
192 push cx
193 push dx
194 push si
195 push di
196 push bp
197 push ds
198 push es
199
200 ; Stack now looks like this:
201 ;
202 ; new [sp + 24]
203 ; flags [sp + 22]
204 ; cs [sp + 20]
205 ; ip [sp + 18]
206 ; ax [sp + 16]
207 ; bx [sp + 14]
208 ; cx [sp + 12]
209 ; dx [sp + 10]
210 ; si [sp + 8]
211 ; di [sp + 6]
212 ; bp [sp + 4]
213 ; ds [sp + 2]
214 ; es [sp] (top of stack)
215
216 mov bx, WORD PTR __T_active
217 mov WORD PTR [bx+2], ss
218 mov WORD PTR [bx], sp
219
220 mov bx, sp
221 mov bx, WORD PTR [bx+24] ; bx = new
222
223 shazam: ; __t_shazam and __t_install
224 ; come here to do the swap

```



```

225|
226|     mov     WORD PTR T_active,bx ; T_active = new;
227|     mov     ss,WORD PTR [bx+2]   ; Switch to new task's stack
228|     mov     sp,WORD PTR [bx]
229|
230|     pop     es
231|     pop     ds
232|     pop     bp
233|     pop     di
234|     pop     si
235|     pop     dx
236|     pop     cx
237|     pop     bx
238|     pop     ax
239|
240|     call    _t_release
241|     sti
242|     iret
243|
244|     _t_swap_in ENDP
245|
246|
247| ;-----
248| ;   _t_chg_chkstk()           Normal stack checking off
249| ;   _t_rst_chkstk()          back on again
250| ;
251| ;   _t_sus_chkstk()          Suspend stack checking temporarily
252| ;   _t_rst_chkstk()          restore it again.
253| ;
254| ; Turn off Microsoft stack checking by overwriting the first
255| ; 5 bytes of _chkstk with an absolute jump to mychkstk.
256| ; This is a kludge but I can't get the Microsoft compiler
257| ; to link my own version of _chkstk, even when I use the
258| ; source file that they supply.
259|
260| chg_chkstk PROC NEAR
261|
262|     mov     bx,OFFSET __chkstk
263|
264|     mov     ah,BYTE PTR cs:[bx+0]
265|     mov     BYTE PTR op,ah
266|
267|     mov     ax,WORD PTR cs:[bx+1]
268|     mov     WORD PTR off,ax
269|
270|     mov     ax,WORD PTR cs:[bx+3]
271|     mov     WORD PTR segm,ax
272|     mov     BYTE PTR cs:[bx+0],0EAH ; EA=JMP
273|     mov     WORD PTR cs:[bx+1],OFFSET mychkstk ; offset
274|     mov     WORD PTR cs:[bx+3],cs ; segment
275|
276|     mov     cs:chk_on,1 ; Enable stack checking
277|     ret
278|
279| chg_chkstk ENDP
280|
281| rst_chkstk PROC NEAR
282|
283|     mov     bx,OFFSET __chkstk
284|
285|     mov     ah,BYTE PTR op
286|     mov     BYTE PTR cs:[bx+0],ah
287|
288|     mov     ax,WORD PTR off
289|     mov     WORD PTR cs:[bx+1],ax
290|
291|     mov     ax,WORD PTR segm
292|     mov     WORD PTR cs:[bx+3],ax
293|     ret
294|
295| rst_chkstk ENDP
296|
297| _t_sus_chkstk PROC NEAR
298|     mov     cs:chk_on,0
299|     ret
300|
301| _t_sus_chkstk ENDP
302|
303| _t_rst_chkstk PROC NEAR
304|     mov     cs:chk_on,1
305|     ret
306|
307| _t_rst_chkstk ENDP
308| ;-----
309| ; On entry AX holds the number of bytes required for local
310| ; variables. Chkstk normally checks the stack and, at the
311| ; same time, finishes setting up the stack frame by
312| ; subtracting the contents of ax from the stack pointer.
313| ;
314| mychkstk PROC NEAR
315|
316|     mov     cx,cs:chk_on ; If stack checking disabled at
317|     or     cx,cx ; run time, skip past the actual
318|     jz     nocheck ; test.
319|
320|     mov     cx,T_active
321|     add     cx,44 ; Offset to stack base + 4
322|     cmp     sp,cx ; if( sp <= stack_base )
323|     jbe     stack_err
324|
325| nocheck:
326|     pop     cx ; cx = return address
327|     sub     sp,ax ; finish setting up stack frame
328|     jmp     cx ; ret to caller w/o modifying stack.
329|
330| stack_err:
331|     mov     ax,-9 ; Return address still on the stack
332|     push    ax
333|     call    _t_stop ; Shouldn't return
334|

```

(continued on next page)

AMX 86

KADAK's
engineers bring
years of practical real-time
experience to this mature

MULTI-TASKING SYSTEM (version 2.0)

for the IBM® PC, PC/XT and PC/AT

- No royalties
- IBM PC DOS® support
- C language support
- Preemptive scheduler
- Time slicing available
- Source code of the C interface and device drivers is included
- Intertask message passing
- Dynamic operations:
 - task create/delete
 - task priorities
 - memory allocation
- Event Manager
- Semaphore Manager

AMX86™ operates on any 8086/88, 80186/88, 80286 system.

Demo package \$25 US
Manual only \$75 US
AMX 86 system \$2195 US
(shipping/handling extra)

KADAK Products Ltd.

206-1847 W. Broadway
Vancouver, B.C., Canada
V6J 1Y5



Telephone: (604) 734-2796
Telex: 04-55670

Also available for 8080, Z80, 68000

CIRCLE 193 ON READER SERVICE CARD

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

FEATURING

- Complete Amiga Hardware and Software reviews
- A vast and growing library of over 110 PDS Disks
- Solid and informative for both the advanced and beginning Amiga User
- Understandable program listings and tools
- Step by Step Hardware projects

Amiga Users have made Amazing Computing™ the longest running Monthly magazine dedicated to the Commodore Amiga. If you are searching for Amiga technical information that is both current and comprehensive, then be amazed by the pioneer Amiga Magazine, Amazing Computing - your Original AMIGA Monthly Resource.

YES, Amaze Me! I have enclosed \$24.00 U.S. (\$30.00 Canada & Mexico, \$35.00 Overseas) in check or money order (U.S. funds drawn on a U.S. bank) to:

PiM Publications, Inc.
P.O. Box 869-DD
Fall River, MA 02722

Name _____
Address _____
City _____ St _____ Zip _____

CIRCLE 194 ON READER SERVICE CARD

Listing Three

(Listing continued, text begins on page 126.)

```

335|    mov cx,0
336|    jmp cx          ; Panic abort to DOS
337|
338| mychkstk ENDP
339|
340| TEXT    ENDS
341| END

```

End Listing Three

Listing Four

Listing 4 -- globals.c, Printed 9/11/1987

```

1| #define ALLOC 1
2|
3| #include "kernel.h"

```

End Listing Four

Listing Five

Listing 5 -- queue.c, Printed 9/11/1987

```

1| #include "kernel.h"
2|
3| /*-----*/
4|
5| T_QUEUE *t_makequeue( size )
6| int      size;
7| {
8|     /* Make a queue for intertask communication and link it
9|      * into the queue chain. Return values are pointer
10|      * to queue on success or TE_NOMEM if insufficient memory.
11|      * Queues are searched in order of creation so it's best
12|      * to create the most active queues first. Be sure that
13|      * multitasking is blocked when tailp is modified (because
14|      * it's static).
15|      */
16|
17|     static T_QUEUE *tailp;
18|     T_QUEUE *p;
19|     void *m;
20|
21|     t_block();
22|     p = (T_QUEUE *) malloc( sizeof(T_QUEUE)
23|                             + ((size-1) * sizeof(void *)) );
24|
25|     t_release();
26|
27|     if( !p )
28|         return (T_QUEUE *) TE_NOMEM;
29|
30|     p->signature = TQ_SIG;
31|     p->next = NULL;
32|     p->task_h = NULL;
33|     p->task_t = NULL;
34|     p->numele = 0;
35|     p->q_size = size;
36|     p->headp = p->queue;
37|     p->tailp = p->queue;
38|
39|     t_block();
40|
41|     if( !T_queues )
42|         T_queues = tailp = p;
43|     else
44|     {
45|         tailp->next = p;
46|         tailp = p;
47|     }
48|
49|     t_release();
50|     return p;
51| }
52|
53| /*-----*/
54|
55| int t_send( q, msg )
56| T_QUEUE *q; /* Pointer to queue */
57| void *msg; /* Pointer to message to enqueue */
58| {
59|     /* Send a message and reschedule if necessary.
60|      *
61|      * Return Values:
62|      *   TE_NOERR      No error;
63|      *   TE_BADARG     Bad q argument.
64|      *   TE_QFULL      Queue is full
65|      *
66|      * The message is always enqueued in the indicated queue.
67|      * Then, if a task is waiting, The message at the head of
68|      * the queue is dequeued and attached to the task, which
69|      * is put back into the active list. Finally, if a task
70|      * was activated, the current process yields. Note,
71|      * however, that the current task will still be the
72|      * active task if it's higher priority than the one to
73|      * which you send a message. The sending task should wait()
74|      * somewhere to make room for the lower-priority task.

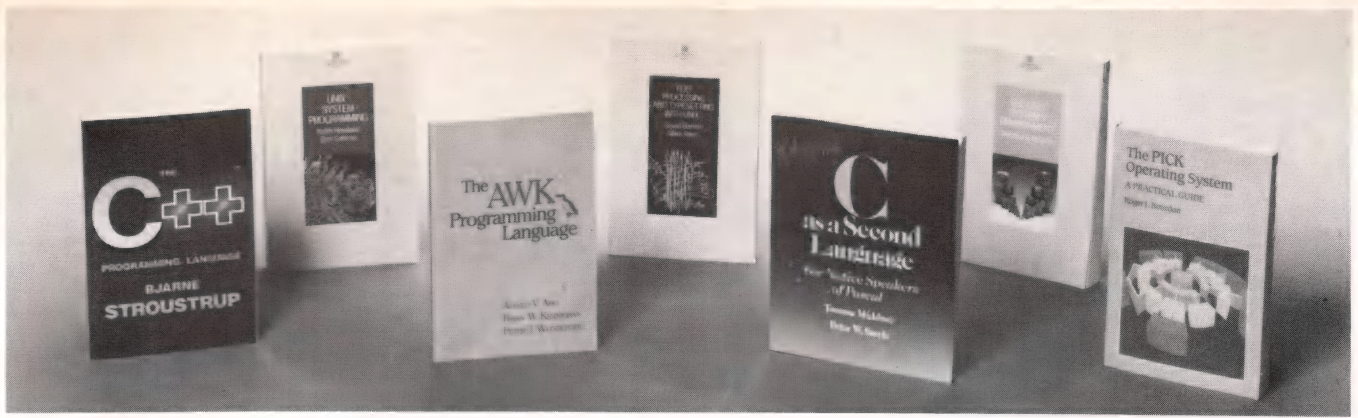
```

```

74|     */
75|
76|     TCB *task;
77|
78|     if( q->signature != TQ_SIG )
79|         return( TE_BADARG );
80|
81|     t_block();
82|
83|     if( q->numele == q->q_size ) /* Queue is full */
84|     {
85|         t_release();
86|         return( TE_QFULL );
87|     }
88|
89|     /*
90|      * Enqueue the message.
91|      */
92|
93|     ++q->numele;
94|     if( ++q->tailp >= q->queue + q->q_size )
95|         q->tailp = q->queue;
96|     *(q->tailp) = msg;
97|
98|
99|     if( q->task_h )
100|     {
101|         /* A task is waiting, dequeue both it and the message,
102|          * attach the message to the task, and reschedule
103|          */
104|
105|         task = q->task_h;
106|         q->task_h = task->next;
107|
108|         --q->numele;
109|         if( ++q->headp >= q->queue + q->q_size )
110|             q->headp = q->queue;
111|
112|         task->msg = *(q->headp);
113|         task->status = TS_WAIT;
114|         pq_ins( T_tasks, task );
115|
116|         t_yield();
117|     }
118|
119|     t_release();
120|     return TE_NOERR;
121| }
122|
123| /*-----*/
124|
125| void *t_wait( q, timeout )
126| T_QUEUE *q;
127| int      timeout;
128| {
129|     /* Wait for a message to arrive at the queue. If several
130|      * tasks are waiting at the same queue, the first task
131|      * in the queue gets the message. Return if no message
132|      * arrives within timeout system clock ticks. Maximum
133|      * timeout is 32,767 ticks. A 0 timeout value
134|      * means that the subroutine returns immediately
135|      * (without a reschedule) if no message is waiting
136|      * in the queue.
137|
138|      * Message requests are queued up in order recieved,
139|      * without regard to priority. I've done this both
140|      * because it's easy and because, in most applications,
141|      * tasks with different priorities will not be pending
142|      * on the same queue.
143|
144|      * If a message is present, the routine returns it
145|      * immediately without yielding, otherwise the current
146|      * task is removed from the active list and yield()
147|      * is called.
148|
149|      * Hints: Use this routine to suspend a task for a
150|      * limited amount of time (as compared to deleting
151|      * the task). Just pend on a queue that will never have
152|      * a message sent to it.
153|
154|      * Normally, a pointer to the message is returned, other
155|      * return values are:
156|
157|      *   TE_TIMEOUT on a timeout or if the input value of
158|      *   timeout is 0 and no message is waiting
159|
160|      *   TE_NOTASKS There current task is the only one in
161|      *   existence. This is a guaranteed deadlock.
162|      */
163|
164|     TCB *new;
165|
166|     if( q->signature != TQ_SIG )
167|         return( TE_BADARG );
168|
169|     t_block();
170|
171|     if( q->numele )
172|     {
173|         /* There's a message waiting in the queue. Dequeue
174|          * the message and return it immediately. Strictly
175|          * speaking, we don't have to attach the message
176|          * to the task, but it's convenient to do it for
177|          * debugging reasons.
178|          */
179|
180|         --q->numele;
181|         if( ++q->headp >= q->queue + q->q_size )
182|             q->headp = q->queue;
183|
184|         T_active->msg = *(q->headp);
185|         T_active->status = TS_WAIT;
186|
187|         t_release();

```

(continued on page 120)



All systems are go!

When you get your operating systems books
from the industry leader.

- ☐ **C As a Second Language for Native Speakers of Pascal**—Tomasz Müldner and Peter Steele, both of Acadia University

Here's a complete and detailed description of the C programming language and programming techniques in C. (19210) 456 pp. Paper 1987 \$26.95

- ☐ **The C Book**—Michael Banahan, *The Instruction Set, Ltd.*
This tutorial, based on Banahan's training courses, features a structured approach to C. (17370) 304 pp. Paper 1988 \$24.75*

*Order this book today, and we'll ship it to you as soon as it's available (January '88)!

- ☐ **The C++ Programming Language**—Bjarne Stroustrup, AT&T Bell Laboratories

The definitive reference to the C++ language, an extension of C that Stroustrup developed at AT&T Bell Laboratories. (12078) 328 pp. Paper 1986 \$29.25

- ☐ **The PICK Operating System: A Practical Guide**—Roger J. Bourdon, *Aston Technology Ltd.*

Here's a solid introduction to the management of a PICK-based computer system, covering all the major features of PICK. (18055) 450 pp. Paper 1987 \$32.25

- ☐ **The UNIX® System V Environment**—Stephen R. Bourne, *Digital Equipment Corporation*

Written by a key member of the group that developed the UNIX system, here's a gold mine of information on UNIX toolkits, along with a definitive reference to the new standard, System V. (18484) 378 pp. Paper 1986 \$26.95

- ☐ **Text Processing and Typesetting with UNIX**—David Barron, *University of Southampton*, and Michael Rees, *University of Tasmania*

Written in a clear tutorial style, this book offers a complete description of the capabilities of UNIX text processing and typesetting for a wide audience. (14219) 250 pp. Paper 1987 \$26.95

- ☐ **UNIX System Programming**—Keith Haviland and Ben Salama, both of *Sphinx Ltd.*

Here's an advanced programming book for software developers producing applications written in C for the UNIX system. (12919) 480 pp. Paper 1987 \$26.95

- ☐ **Operating Systems: A Systematic View, Third Edition**—William S. Davis, *Miami University, Ohio*

An update of the popular book that gives you a straightforward introduction to operating systems. (11185) 539 pp. Hardcover 1987 \$37.75

- ☐ **An Introduction to Operating Systems, Revised First Edition**—Harvey M. Deitel, *Boston College*

In addition to complete coverage of the fundamental concepts of operating systems, this best-seller includes five case studies—on UNIX, VAX/VMS, CP/M, MVS, and VM. (14501) 673 pp. Hardcover 1984 \$44.25

- ☐ **Operating Systems Concepts, Second Edition**—James L. Peterson and Abraham Silberschatz, both of the *University of Texas at Austin*

This popular book clearly and concisely defines the fundamental concepts of operating systems and their application to any system. (06198) 625 pp. Hardcover 1985 \$44.25

OF SPECIAL INTEREST...

- ☐ **The AWK Programming Language**—Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, all of *AT&T Bell Laboratories*

Here's the authoritative reference on the newest version of AWK, a pattern-matching language for writing short programs to perform common data-manipulation tasks. Written by AWK's original developers, the book begins with a tutorial that shows you how easy AWK is to use, and continues with a comprehensive manual. (07981) 210 pp. Paper 1987 \$23.75



UNIX is a registered trademark of AT&T.

To order, check off the books that interest you, tear out this ad and send check for total, plus your local sales tax, to: Dept. DM, Addison-Wesley Publishing Co., Reading, MA, 01867-9984. Or simply call the number below.

◆ **Addison-Wesley Publishing Company**
Reading, Massachusetts 01867 • (617) 944-3700
CIRCLE 195 ON READER SERVICE CARD

Listing Five

(Listing continued, text begins on page 126.)

```

188:         return T_active->msg;
189:     }
190:     else
191:     {
192:         /* No messages waiting */
193:         if( timeout == 0 )
194:         {
195:             /* Immediate time out */
196:             t_release();
197:             return TE_TIMEOUT;
198:         }
199:         else
200:         {
201:             /* Enqueue the current task to wait for an
202:              * incoming message. The pq_del call
203:              * gets a task to preempt the current one.
204:              */
205:             if( !pq_del( T_tasks, &new ) )
206:             {
207:                 t_release();
208:                 return TE_NOTASKS;
209:             }
210:             else
211:             {
212:                 T_active->wait = timeout;
213:                 T_active->next = NULL;
214:                 T_active->timestamp = T_clock;
215:             }
216:             if( !q->task_h )
217:                 q->task_h = T_active;
218:             else
219:                 q->task_t->next = T_active;
220:             q->task_t = T_active;
221:             _t_swap_in( new ); /* Returns on either a message
222:                               * being */
223:                               /* sent to this queue or a
224:                               * timeout. */
225:             return ( T_active->msg ? T_active->msg : TE_TIMEOUT );
226:         }
227:     }
228: }
229: }
230: }
231: }
232: }
233: }
234: }
235: }
236: }
237: }
238: }
239: }
240: }
241: }
242: }
243: }
244: }
245: }
246: }
247: }
248: }
249: }
250: }
251: }
252: }
253: }
254: }
255: }
256: }
257: }
258: }
259: }
260: }
261: }
262: }

```

End Listing Five

Listing Six

Listing 6 -- task.c, Printed 9/11/1987

```

1: #include <dos.h>
2: #include <stdarg.h>
3: #include <signal.h>
4: #include <tools/hardware.h> /* #define for TIMR_CLK */
5: #include <kernel.h>
6:
7: static int Speedup_factor = 0;
8: static long Executed, Timed_out, Did_swap;
9:
10: #define max(a,b) ((a) > (b) ? (a) : (b))
11:
12: /*-----
13:  * Strings for error codes. Note that these are upside down
14:  * to compensate for the negative indexes
15:  */
16:
17: static char *msgs[] =
18: {
19:     "Ctrl-Break or Ctrl-C"

```

/* -10 */

```

20: "stack overflow", /* -9 */
21: "delete would have caused a deadlock", /* -8 */
22: "internal error", /* -7 */
23: "No tasks to send message to", /* -6 */
24: "Queue is full", /* -5 */
25: "Timeout", /* -4 */
26: "Illegal Argument", /* -3 */
27: "Insufficient memory available", /* -2 */
28: "Maximum number of tasks (32) already exists", /* -1 */
29: "No error" /* 0 */
30: };
31:
32: char **errlist = msgs + ((sizeof(msgs)/sizeof(*msgs)) - 1);
33:
34: /*-----
35:  * t_iserr(x)
36:  * {
37:  *     return( -10 <= (x) && (x) < 0 );
38:  * }
39:  */
40:
41: /*-----
42:  * t_perror( str, errcode )
43:  * char *str;
44:  * {
45:  *     if( !t_iserr(errcode) && errcode != 0 )
46:  *         t_printf( "%s status %d\n", str, errcode );
47:  *     else
48:  *         t_printf( "%s %s\n", str, errlist[errcode] );
49:  * }
50:  */
51:
52: /*-----
53:  * static intr()
54:  * {
55:  *     t_stop( TE_KILL );
56:  * }
57:  */
58:
59: /*-----
60:  * t_start( speedup_factor )
61:  * {
62:  *     /* Start multitasking. At least one task must have
63:  *      * been created prior to this call. The speedup factor
64:  *      * determines the system clock-tick rate. A value of 1
65:  *      * gives the default rate of roughly 18.2 times a second
66:  *      * (once every 55 milliseconds, more or less). A
67:  *      * speedup factor of 2 gives twice that speed: 36.4 ticks
68:  *      * per second, one every 27 milliseconds or thereabouts.
69:  *      * Speeding up the clock rate shouldn't affect the DOS
70:  *      * clock. Nonetheless, it's safest if the speedup factor
71:  *      * is a power of two.
72:  *      *
73:  *      * If the speedup factor is 0 then the system is
74:  *      * nonpreemptive. You'll have to use t_yield(), t_send(),
75:  *      * and t_wait() to change contexts.
76:  *      *
77:  *      * Control passes immediately to the highest priority task.
78:  *      * Control will automatically pass back to the calling
79:  *      * subroutine when all tasks have been deleted. The task
80:  *      * is actually started in _t_shazam(), declared in swap.asm.
81:  *      *
82:  *      * Normal return values:
83:  *      *
84:  *      * TE_NOTASKS No tasks exist, multitasking not
85:  *      * started;
86:  *      *
87:  *      * TE_DEADLOCK A task deleted itself and it's the
88:  *      * only active task in the system. Other
89:  *      * tasks exist but they're all pending
90:  *      * on queues.
91:  *      *
92:  *      * TE_NOERR All tasks have been deleted normally,
93:  *      * no tasks are waiting on queues.
94:  *      *
95:  *      * TE_STACK Task stack overflow.
96:  *      *
97:  *      * If TE_NOERR is returned, then all memory allocated to
98:  *      * tasks will have been saved, otherwise, if one of the
99:  *      * above errors was returned, T_active will point at the
100:  *      * TCB of the offending task.
101:  *      *
102:  *      * Other return values are possible if a task calls t_stop()
103:  *      * directly. The argument passed to t_stop() is returned by
104:  *      * t_start(). The process is analogous to the value of
105:  *      * exit(), which doesn't return and who's argument is
106:  *      * passed back to a wait() call in the parent process. Note
107:  *      * that the TCB pointed to by T_active will not be free()'ed
108:  *      * unless TE_NOERR (0) is returned.
109:  *      */
110:
111:     Speedup_factor = speedup_factor;
112:
113:     if( !pq_del( T_tasks, &T_active ) )
114:         return TE_NOTASKS;
115:
116:     if( speedup_factor > 0 )
117:     {
118:         t_cli();
119:         _t_speedup( speedup_factor );
120:     }
121:
122:     signal( SIGINT, intr );
123:
124:     t_shazam();
125:     return TE_INTERNAL; /* Shouldn't ever get here */
126: }
127:
128: /*-----
129:  * t_second()
130:  * {
131:  *     int
132:  *     {
133:  *         /* Returns the number of system clock ticks in a

```



```

134|      * second, given the speedup factor passed to t_start().
135|      * Returns 0 if the speedup factor was 0. In this case
136|      * a t_wait() call will never time out.
137|      */
138|
139|      return (TIMR_CLK / 65536) * Speedup_factor ;
140|  }
141|  /-----*/
142|  TCB *t_create( subr, tag, priority, stack_size, ... )
143|  {
144|      /* Subroutine that forms main module */
145|      int (*subr)(); /* Subroutine used to identify TCB */
146|      char *tag; /* Priority */
147|      unsigned priority; /* Stack size (in 2-byte words) */
148|      int stack_size; /* Stack size (in 2-byte words) */
149|      {
150|          /* Creates a new task. Subr is a pointer to the main()
151|           * subroutine for the task.
152|           *
153|           * Priorities must be in the range 0-255. 255 is the
154|           * highest. If more than one task has the same priority,
155|           * they are executed in a round-robin
156|           * fashion. Forces a reschedule if tasking is active.
157|           *
158|           * Arguments may be passed to the subroutine at startup.
159|           * That is, a NULL-terminated list of pointer-sized
160|           * arguments follow stack size in the t_create() call.
161|           * These are passed to the subroutine in the normal way.
162|           * For example:
163|           *
164|           * foo( a, b, c ) int a, b, c; {}
165|           * t_create( foo, "foo" 10, 128, doo, wha, ditty, NULL);
166|           *
167|           * starts up foo() as a task at priority 10 with a
168|           * 128-byte stack. Doo, wha, and ditty are passed
169|           * to foo as arguments a, b, and c. Note that the
170|           * arguments use 6 of the 128 bytes in the stack
171|           * (two for each argument). The "foo" tag is just
172|           * used for identification purposes in debugging.
173|           * It can be any string.
174|           *
175|           * Note that a few Microsoft functions (like printf)
176|           * use up inordinate amounts of stack. If you're going
177|           * to call Microsoft library routines, you'll need
178|           * at least 1K bytes of stack (stack_size==512) per
179|           * task.
180|           *
181|           * A pointer to the created TCB is returned normally.
182|           * Error return values are:
183|           *
184|           * TE_TOO_MANY Maximum number of tasks already exists
185|           * TE_NO_MEM Insufficient memory available
186|           */
187|      }
188|
189|      TCB *t;
190|      struct SREGS segs;
191|      int pq_cmp();
192|      int pq_swap();
193|      va_list argptr;
194|      void *arg;
195|      void *malloc();
196|
197|      va_start( argptr, stack_size );
198|
199|      if( ++T_numtasks > T_MAXTASK )
200|          return( TCB *) TE_TOO_MANY;
201|
202|      t_block();
203|
204|      /* Allocate the stack, converting stack size to bytes.
205|       * I'm requesting one more cell than specified in order
206|       * to make room for the error return address, below
207|       * [stack[0] is included in the sizeof(TCB)]. The minimum
208|       * stack size is 20 words, this gives us enough for a
209|       * context swap plus a little slop.
210|       */
211|
212|      stack_size = max( stack_size, 20 );
213|
214|      if( !(t = (TCB *) malloc(sizeof(TCB)+(stack_size*sizeof(void*)))
215|          {
216|              t_release();
217|              return( TCB *) TE_NO_MEM;
218|          }
219|
220|      /* Create the active queue if necessary, then initialize
221|       * the TCB. The stack pointer is initialized to point
222|       * just past the end of the stack (rather than to the
223|       * last cell) because a push uses a predecrement. The
224|       * PC points at the subroutine. Uninitialized registers
225|       * are unimportant, but will contain 0. The stack area
226|       * is initialized to the pattern a5a5a5a5... so that
227|       * we can look it with a debugger and see what's been
228|       * used. 0 is no good for this purpose because 0 is
229|       * a likely thing to be pushed on the stack.
230|       */
231|
232|      if( !T_tasks )
233|          T_tasks = pq_create( T_MAXTASK, sizeof(TCB *),
234|                              pq_cmp, pq_swap, NULL );
235|      segread( &segs );
236|
237|      memset( t, 0x0, sizeof(TCB) - sizeof(void*) );
238|      memset( t->stack, 0xa5, stack_size * sizeof(void*) );
239|
240|      t->sp = t->stack + ++stack_size;
241|      t->ds = segs.ds; /* Stacks are in data seg */
242|      t->initial_sp = t->sp;
243|      t->priority = priority;
244|      t->timestamp = T_clock;
245|      t->tag = tag;
246|

```

(continued on next page)

Step into the Future

Stony Brook Introducing **MODULA-2** for 8086 family machines



Some day, the other
Modula-2 compilers
might have what we
have NOW:

SPEED

- Compiles 5K lines / minute on PC/AT
- Runs Sieve faster than Microsoft C V4.0

FLEXIBILITY

- Generates Microsoft standard objects
- Supports 6 memory models + mixed model
- Interfaces directly to other DOS languages (you don't have to throw out your C code!)

POWER

- Full Modula-2 language
- + Array and record constants
- + Substrings and sub-arrays
- + Tailorable procedure calling

REPERTOIRE by PMI,
indexed file and
screen management
package included free!

PLUS OS/2 and Microsoft NOW Windows compatible

The generated code is compatible with OS/2 and Microsoft Windows. Runtime libraries will be available soon.

Stony Brook
INC
SOFTWARE

Forest Road
Wilton, New Hampshire 03086
(603) 654-2525

Compiler and DOS runtime
library objects only:

\$195

Above plus editor, source
debug, make utility, and
runtime library sources:

\$345

No other
Modula-2
compiler
comes close!

Add \$5 shipping and handling in North America, \$15 for over-seas orders. VISA and MasterCard accepted.

CIRCLE 196 ON READER SERVICE CARD

Listing Six

(Listing continued, text begins on page 126.)

```

247  /* Initialize the stack, First pretend that we've
248  * already called the initial subroutine by pushing
249  * the arguments, and t_stop as a dummy return address.
250  * The task shouldn't return, but if it does, t_stop
251  * seems like a reasonable thing to call, even though
252  * it will return garbage.
253  * The last 11 things are the initial context.
254  * They'll be popped as part of the context swap.
255  */
256
257  while( arg = va_arg(argptr, void*) )
258  {
259      *--(t->sp) = arg;
260
261      *--(t->sp) = t_stop; /* Vector to t_stop */
262      *--(t->sp) = 0; /* flags */
263      *--(t->sp) = segs.cs; /* cs */
264      *--(t->sp) = subr; /* ip */
265      *--(t->sp) = 0; /* ax */
266      *--(t->sp) = 0; /* bx */
267      *--(t->sp) = 0; /* cx */
268      *--(t->sp) = 0; /* dx */
269      *--(t->sp) = 0; /* si */
270      *--(t->sp) = 0; /* di */
271      *--(t->sp) = segs.ds; /* bp */
272      *--(t->sp) = segs.es; /* es */
273
274      if( T_active && T_PRIORITY( t, T_active ) <= 0 )
275      {
276          pq_ins( T_tasks, &T_active );
277          t_swap_in( t );
278      }
279      else
280      {
281          pq_ins( T_tasks, &t );
282          t_release();
283      }
284      return t;
285  }
286  }
287  /*-----*/
288
289  static TCB *del_fm_queues( task )
290  TCB *task;
291  {
292      /* Traverse the queue list and if the task is waiting
293      * for a message, delete it from the queue
294      * and return a pointer to it, otherwise return NULL.
295      */
296
297      T_QUEUE *q;
298      TCB *t, **prev;
299
300      for( q = T_queues; q; q = q->next )
301      {
302          prev = &q->task_h;
303          for( t = q->task_h; t; prev = &t->next, t = t->next )
304          {
305              if( t == task )
306              {
307                  *prev = t->next;
308                  return t;
309              }
310          }
311      }
312      return NULL;
313  }
314  /*-----*/
315
316  int t_chg_priority( tp, new_priority )
317  TCB *tp;
318  int new_priority;
319  {
320      /* Change priority for indicated task. Forces a reschedule.
321      * If the task was waiting on a message, it is immediately
322      * timed out and put back on the active list. A task may
323      * change it's own priority.
324      *
325      * Return values:
326      *
327      * TE_NOERR Task doesn't exist;
328      * TE_BADARG
329      */
330
331      int rval = TE_NOERR;
332      TCB *deleted;
333      int pq_rm_cmp();
334
335      if( new_priority > 255 )
336          return TE_BADARG;
337
338      t_block();
339
340      if( tp == T_active )
341      {
342          T_active->priority = new_priority;
343          t_yield();
344      }
345      else
346      {
347          if( !pq_remove( T_tasks, &deleted, pq_rm_cmp, tp ) )
348          {
349              if( deleted == del_fm_queues( tp ) )
350              {
351                  deleted->status = TS_TIMEOUT;
352                  deleted->msg = NULL;
353              }
354              else
355              {
356                  return TE_BADARG;
357              }
358          }
359      }
360
361      deleted->priority = new_priority;
362      pq_ins( T_tasks, &deleted );
363      t_yield();
364  }
365  /*-----*/
366
367  int t_delete( task )
368  TCB *task;
369  {
370      /* Delete task created with previous t_create() call and
371      * free all memory associated with task. Note that
372      * malloced() memory is not freed, only the memory that
373      * t_create() allocated to begin with. A task may delete
374      * itself. Forces a reschedule.
375      *
376      * Return values:
377      *
378      * TE_BADARG Task doesn't exist
379      * TE_NOERR
380      *
381      * T_stop is called automatically when the only active
382      * task deletes itself. See t_start() for an explanation
383      * of the return stat.
384      *
385      * For convenience, a task may delete itself with a
386      * t_delete(NULL) call.
387      */
388
389      TCB *deleted;
390      static TCB garbage;
391      int pq_rm_cmp();
392
393      t_block();
394
395      if( T_active == task || task == NULL )
396      {
397          /* Delete the current task
398          * Note that the t_install() call below will not
399          * return. It replaces the current task with the new
400          * one, a pointer to which is in "deleted."
401          */
402
403          if( !pq_del( T_tasks, &deleted ) )
404              t_stop( T_numtasks <= 1 ? TE_NOERR : TE_DEADLOCK );
405          else
406          {
407              --T_numtasks;
408              _t_install( deleted );
409          }
410      }
411      else
412      {
413          /* Delete a task that's not active. pq_remove tries
414          * to get it from the active list. If that's not
415          * successful, del_fm_queues() scans the queues looking
416          * for it. If that's not successful, TE_BADARG is
417          * returned.
418          */
419
420          if( pq_remove( T_tasks, &deleted, pq_rm_cmp, task ) )
421              free( deleted );
422
423          else if( deleted == del_fm_queues(task) )
424              free( deleted );
425          else
426          {
427              t_release();
428              return TE_BADARG;
429          }
430      }
431
432      if( --T_numtasks <= 0 ) /* Deleted the only task */
433          t_stop( TE_NOERR );
434
435      t_release();
436      return TE_NOERR;
437  }
438  /*-----*/
439
440  static pq_cmp( task1, task2 )
441  TCB **task1, **task2;
442  {
443      return T_PRIORITY( *task1, *task2 );
444  }
445
446  static pq_swap( task1, task2 )
447  TCB **task1, **task2;
448  {
449      TCB *tmp;
450
451      tmp = *task1;
452      *task1 = *task2;
453      *task2 = tmp;
454  }
455
456  static pq_rm_cmp( task1, item )
457  TCB **task1;
458  TCB *item;
459  {
460      return !(*task1 == item);
461  }
462  /*-----*/
463
464  static outc(c)
465  {
466      if( c == '\n' )
467          putchar('\r');
468      putchar(c);
469  }

```



```

473| }
474|
475| t_printf( fmt )
476| char *fmt;
477| {
478|     /* The doprnt() function used here is from: Allen Holub,
479|      * The C Companion (Englewood Cliffs: Prentice-Hall,
480|      * 1987). You can also use the ANSI vprintf(). Note
481|      * that the Microsoft version of vprintf() uses A LOT
482|      * of stack. If you're using vprintf, your tasks should
483|      * have at least 1K bytes of stack.
484|      */
485|
486|     va_list args;
487|
488|     va_start(args, fmt);
489|
490|     t_block();
491|     doprnt(outc, 0, fmt, args); /* vprintf( fmt, args ); */
492|     t_release();
493| }
494|
495| /*-----*/
496|
497| t_stats()
498| {
499|     /* Print various scheduler related statistics. This routine
500|      * should not be called from a task (because it uses printf).
501|      */
502|
503|     printf("\nScheduler called %ld times: %ld Tasks timed out, "
504|            "%ld context swaps\n",
505|            Executed, Timed_out, Did_swap );
506| }
507|
508| /*-----*/
509|
510| #pragma check_stack-
511|
512| TCB *_t_reschedule()
513| {
514|     static T_QUEUE *q;
515|     static TCB *t, **prev;
516|
517|     /* Workhorse function called by the scheduler
518|      * (timer-interrupt service routine).
519|
520|      * Scan all the queues, checking for timed-out tasks. If
521|      * you find one, remove it from the queue and add it to
522|      * the active list.
523|
524|      * Modify T_active to point at the next task to activate.
525|      * (the original T_active if no change).
526|      */
527|
528|     ++Executed;
529|
530|     _t_sus_chkstk(); /* Stack probes off for the nonce */
531|
532|     for( q = T_queues; q; q = q->next )
533|     {
534|         prev = &(q->task_h);
535|
536|         for( t = q->task_h; t; )
537|         {
538|             if( --(t->wait) <= 0 )
539|             {
540|                 ++Timed_out;
541|                 *prev = t->next;
542|                 t->msg = NULL;
543|                 t->status = TE_TIMEOUT;
544|                 pq_ins( T_tasks, &t );
545|             }
546|
547|             prev = &(t->next);
548|             t = t->next;
549|         }
550|
551|         /* Check the highest-priority element of the queue.
552|          * If it's not higher than the current task, do nothing.
553|          * Otherwise do a context swap. pq_replace will
554|          * extract the highest priority object from the active
555|          * list and put it into t, simultaneously putting
556|          * T_active into the list.
557|          */
558|
559|         T_active->timestamp = ++T_clock;
560|
561|         if( t = *( TCB ** ) pq_look( T_tasks ) )
562|         {
563|             if( T_PRIORITY(t, T_active) >= 0 )
564|             {
565|                 ++Did_swap;
566|                 pq_replace( T_tasks, &t, &T_active );
567|                 T_active = t;
568|             }
569|         }
570|
571|         _t_rst_chkstk(); /* Stack probes on again */
572|     }
573| }
574| #pragma check_stack+

```

End Listing Six

Listing Seven

Listing 7 -- tdebug.c, Printed 9/11/1987

```

1| #include <stdio.h>
2| #include "kernel.h"

```

```

3|
4| /* TDEBUG.C Various routines that are useful for debugging but
5|  * probably won't end up in the final system.
6|  */
7|
8| t_tprint( t )
9| TCB *t;
10| {
11|     /* Print a TCB */
12|
13|     char **p, *str;
14|     int i;
15|
16|     printf("----- <ts> at %04x ----- \n",
17|            t->tag, t );
18|
19|     printf("stack %04x: %04x, priority %d, timestamp %d, ",
20|            t->ss, t->sp, t->priority, t->timestamp );
21|
22|     printf(" wait %d, status %d\n", t->wait, t->status );
23|
24|     printf("next %04x, initial sp %04x, msg %04x ",
25|            t->next, t->initial_sp, t->msg );
26|
27|     pstr( t->msg, 10 );
28|
29|     printf("stack[ 0 ] @ %04x = %04x, ",
30|            &(t->stack)[0], (t->stack)[0]);
31|     printf("stack[ 1 ] @ %04x = %04x\n\n",
32|            &(t->stack)[1], (t->stack)[1]);
33|
34|     /* Print the top 15 elements of the stack */
35|
36|     i = 15;
37|     for( p = (char **)t->sp; p < t->initial_sp && --i>0; p++ )
38|     {
39|         printf(" sp[%2d] @ %04x = %04x, ",
40|                (char **)t->sp - p, p, *p);
41|     }
42|
43|     if( i < 0 )
44|         printf("(Stack dump truncated at 15 elements)\n");
45|
46|     printf("----- \n");
47| }
48|
49| /*-----*/
50|
51| static pstr( str, len )
52| char *str;
53| {
54|     /* Print a string with dots instead of nonprinting
55|      * characters
56|      */
57|
58|     putchar('<');
59|
60|     for( len = 32; --len>0 && *str; str++ )
61|         putchar( ' ' <= *str && *str < 0x7f ? *str : '.' );
62|
63|     printf("\n");
64| }
65|
66| /*-----*/
67|
68| t_qprint( q )
69| T_QUEUE *q;
70| {
71|     /* Print out the contents of a queue */
72|
73|     TCB *t;
74|     int i;
75|
76|     if( q->signature != TQ_SIG )
77|     {
78|         printf("Queue is invalid (bad signature)\n");
79|         return;
80|     }
81|
82|     printf("----- Queue at %04x ----- \n", q);
83|     printf("%d/%d messages in queue, next queue at %04x\n",
84|            q->numele, q->q_size, q->next );
85|
86|     printf("Waiting tasks: ");
87|     if( t = q->task_h )
88|         printf("(none)\n");
89|     else
90|     {
91|         for( ; t; t = t->next )
92|         {
93|             printf(" %s, ", t->tag );
94|             if( t == q->task_h )
95|                 printf("(end)\n");
96|         }
97|         printf("\n");
98|     }
99|
100|     printf( "head = %queue[%d], tail = %queue[%d], queue is:\n",
101|            q->headp - q->queue, q->tailp - q->queue );
102|
103|     for( i = 0; i < q->q_size; i++ )
104|     {
105|         printf("queue[%d]: %04x ", i, q->queue[i] );
106|         pstr( q->queue[i], 32 );
107|     }
108|     printf("----- \n");
109| }

```

End Listings

Listing One (Text begins on page 144.)

```
( LOAD screen for DDJ Standard Prelude and String Extension)
( MJT Aug 30 1987 for DDJ December 1987)

( 2 LOAD ( Standard prelude)
  3 LOAD ( Augmented interpretation)
  4 5 THRU ( Controlled words)
  6 13 THRU ( Strings)

( FORTH-83 functions-- typical definitions)
( Adjust these words for your Forth. See DDJ Oct 1987.)
( Note: functions already provided need not be redefined.)
: RECURSE [COMPILE] MYSELF ; IMMEDIATE
: INTERPRET INTERPRET ;

: I> ( - 'data) COMPILE R> ; IMMEDIATE
: >I ( - 'data) COMPILE >R ; IMMEDIATE

( Used for alignment: )
: ALIGN ( HERE 1 AND ALLOT) ;
: REALIGN ( a - a' ) ( DUP 1 AND +) ;

2 CONSTANT CELL : CELL+ 2+ ; : CELLS 2* ;

: UNDO I> R> R> 2DROP >I ; \ Undoes a DO-- LOOP.
( Required definitions - used to support further compilation)

: THRU ( n n2) 1+ SWAP DO I LOAD LOOP ;
\ LOADS screens n through n2.

: \ >IN @ 64 + -64 AND >IN ! ; IMMEDIATE
\ comment to end of line. For use in screens only.

: \ 1024 >IN ! ; IMMEDIATE
\ stops interpreting or compiling screen immediately.

: \IF ( f ) 0= IF [COMPILE] \ THEN ; IMMEDIATE
\ conditional interpretation or compilation.

: NEED ( - f ) 32 ( ie blank) WORD FIND SWAP DROP 0= ;
\ true if the following word is in the search order.
\ FORTH-83 Controlled Words

NEED 2* \IF : 2* DUP + ;
NEED D2* \IF : D2* 2DUP D+ ;

NEED HEX \IF : HEX 16 BASE ! ;
NEED C, \IF : C, ( n ) HERE 1 ALLOT C! ;

NEED BL \IF 32 CONSTANT BL

NEED ERASE \IF : ERASE ( a n ) 00 FILL ;
NEED BLANK \IF : BLANK ( a n ) BL FILL ;

NEED .R \IF : .R ( n width) >R DUP 0< R> D.R ;

\ DDJ Forth Column Controlled Words
NEED 2>R
\IF : 2>R COMPILE SWAP COMPILE >R COMPILE >R ; IMMEDIATE
NEED 2R>
\IF : 2R> COMPILE R> COMPILE R> COMPILE SWAP ; IMMEDIATE
NEED @EXECUTE \IF : @EXECUTE @ EXECUTE ;

NEED AGAIN
\IF : AGAIN 0 [COMPILE] LITERAL [COMPILE] UNTIL ; IMMEDIATE
NEED DLITERAL
DUP \IF : DLITERAL SWAP [COMPILE] LITERAL [COMPILE] LITERAL ;
\IF IMMEDIATE

NEED S>D \IF : S>D ( n - d) DUP 0< ;
NEED WITHIN \IF : WITHIN ( n n2 n3 - f) OVER - >R - R> U< ;
NEED TRUE \IF -1 CONSTANT TRUE
\ String primitives

: /STRING ( a n n2 - a+n2 n-n2) ROT OVER + ROT ROT - ;
\ truncates leftmost n chars of string. n may be negative.

VARIABLE CTEMP

: CTO"" ( c - a 1) CTEMP C! CTEMP 1 ;
\ converts character to string.

\ SKIP and SCAN

: SKIP ( a 1 c - a2 12)
\ returns shorter string from first position unequal to byte.
>R BEGIN DUP
WHILE OVER C@ R@ - IF R> DROP EXIT THEN 1 /STRING
REPEAT R> DROP ;
```


Dbase*

programming tools

*Clipper, FoxBASE+,
dBASE, QuickSilver

The UI Programmer

UI is the first professional code generator; we wrote UI for programmers who want to automate their work but cannot use code that is 'almost' good enough. If your user interfaces include bounce-bar menus, pop-up help screens and the other features of today's best programs, you will gain an order of magnitude in productivity with UI.

UI is a second generation, programmable product — so your code comes out your way. Application specific edits, for instance, can be placed in the UI 'template' which controls the generation. Edit the screen appearance until it 'looks and feels' perfect. Everytime you generate code, your special logic is preserved.

Speaking of editing the screen, UI includes a powerful, 3-D screen editor, so you can draw pop-up help boxes over your pull-down menus, over your application.

The Documentor

To run Doc, you just tell it the name of the main-line routine and make sure your printer has a lot of paper! (Sure, you can have the output go to the screen or a file, too.)

You can tailor your documentation to include any or all of: a table of contents, system tree diagram (main line is the root), hierarchy (box diagram) charts for each module, action diagrams (modern style flow charts) for each PRG or procedure, DBF listings (structure, indexes, more), where used/updated listings for fields and all variables — by module and by line number within each module.

Our written money-back satisfaction guarantee set a new standard when we began it in 1985. (Return rate to date: 0.6% and dropping!) No copy protection, royalties or other nonsense.

Suggested retail: \$295 each, (800) support included. At your dealer today. Call us for a very special offer on our latest release! (800) 233-3569 or, in NY, (212) 406-7026.

WallSoft

The Computer Aided Software
Engineering Corporation

233 Broadway, Suite 869, New York, NY 10279
CIRCLE 197 ON READER SERVICE CARD

```
: SCAN ( a 1 byte - a2 12)
\ returns shorter string from first position equal to byte.
>R BEGIN DUP
  WHILE OVER C@ R@ = IF R> DROP EXIT THEN 1 /STRING
  REPEAT R> DROP ;

\ String compilation

: PLACE ( a n a2) 2DUP ! 1+ SWAP CMOVE ;
\ moves string ( a n ) to be a packed string at a2.

: ASCII ( - c) \ value of following character.
  BL WORD 1+ C@ STATE @ \ STATE-smart ASCII
  IF [COMPILE] LITERAL THEN ; IMMEDIATE

: , " \ compiles following string as packed string at HERE
  , ASCII " WORD COUNT DUP >R HERE PLACE R> 1+ ALLOT ALIGN ;
```

\ String literals

```
: ( " ) I> COUNT 2DUP + >I ;

: " ( - a n) STATE @ \ string literal.
  IF COMPILE ( " ) , "
  ELSE ASCII " WORD COUNT >R PAD I CMOVE PAD R> THEN ;
IMMEDIATE
```

\ Number conversion operator

VARIABLE DPL \ punctuation locator.

```
: VAL? ( a n - d 2 , n2 1 , 0)
\ string to number conversion primitive. True if d is valid.
\ Returns d if number contains ",-./:" and sets DPL = 0
\ Returns n if no punctuation present and sets DPL = 0<
  PAD OVER - SWAP OVER >R CMOVE
  BL PAD C! PAD DPL ! 0 0 R> DUP C@ ASCII - = DUP >R - 1-
  BEGIN CONVERT DUP C@ DUP ASCII : =
  SWAP ASCII , ASCII / 1+ WITHIN OR
  WHILE DUP DPL ! REPEAT R> SWAP >R IF DNEGATE THEN
  PAD 1- DPL @ - DPL ! R> PAD = ( valid?)
  IF DPL @ 0< IF DROP 1 ELSE 2 THEN ELSE 2DROP 0 THEN ;
```

\ -TEXT and COMPARE

```
: -TEXT ( a n a2 - -1 , 0 , 1)
\ returns -1 if string a n < a2 n , 0 if equal, and 1 if >.
\ OVER 0= IF ROT 2DROP EXIT THEN
  SWAP 0 DO OVER C@ OVER C@ - ( these chars <> ?)
  IF UNDO C@ SWAP C@ > 2* 1+ EXIT THEN 1 1 D+
  LOOP 2DROP 0 ;

: COMPARE ( a n a2 n2 - -1 , 0 , 1)
\ returns -1 if a n < a2 n2 , 0 if equal, and 1 if >.
  ROT 2DUP ( lengths ) 2>R MIN SWAP -TEXT DUP
  IF 2R> 2DROP
  ELSE DROP 2R> 2DUP = ( lengths = ?)
  IF 2DROP 0 ELSE > 2* 1+ THEN
  THEN ;
\ IN

: -MATCH ( a n a2 n2 - ??? -1 , offset 0)
\ returns the position of string a2 n2 in ( a n ).
\ Offset is zero if ( a n ) is found in first char position.
\ Returns true with invalid offset if ( a n ) isn't in a2 n2.
  2SWAP 2 PICK DUP ( len1 ) >R OVER SWAP - DUP 0< R> 0= OR
  IF 2DROP 2DROP TRUE EXIT THEN
  0 TRUE ( index match? ) ROT 1+ 0
  DO DROP ( index ) >R
  2OVER 2OVER DROP -TEXT 0= ( equal? )
  IF R> 0 LEAVE THEN 1 /STRING R> 1+ TRUE
  LOOP
  2>R 2DROP 2DROP 2R> ;
```

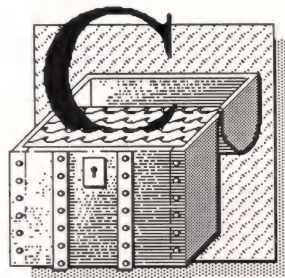
\ Useful string operators

```
: VAL ( a n - d f) VAL? DUP 3 < AND
\ converts string to double number. True if number is valid.
  DUP IF 1 = IF S>D THEN TRUE EXIT THEN DUP DUP ;

: EVAL ( a n )
\ evaluates ("text interprets") a string.
  DUP >R TIB SWAP CMOVE R@ #TIB !
  0 >IN ! 0 BLK ! INTERPRET R> >IN ! ;
```

End Listings

A Preemptive Multitasking Kernel and More Mean Subroutines



The main topic of both this and next month's columns is a small preemptive multitasking kernel for the IBM PC (see Listings One–Seven, beginning on page 110). Most of the kernel is written in C, so it shouldn't be too difficult to port it to another environment, including a ROMed environment. The kernel lets you do multitasking within a program—that is, it lets you run subroutines as independent tasks. It also supports a message-passing system that allows for inter-task communication (and time-outs when a message isn't received within a specified time). It doesn't let you multitask programs, however, and it uses DOS as its I/O system. This month I'll show you how it works; next month I'll dissect the code itself.

This month, I'm also passing on a neat routine for computing a running mean, sent in by Kevin Jennings.

What is Multitasking?

Let's start with some definitions, just so everybody is starting from the same base.

An operating system is a collection of programs that lets you execute other programs. Typically, some of these programs are resident (they stay in the computer's memory), others are nonresident (they stay on the disk until they're needed, whereupon they're read into main memory and executed), and some are somewhere in be-

kernel is in charge of actually reading and executing other programs; the I/O system takes care of all communications with the outside world—the disk, printers, terminals, and so forth; and the shell is the part that talks to you. The shell typically uses both the I/O system and the kernel. This month's column does not present an operating system. In fact it only looks at a small part of the kernel—the part that takes care of multitasking.

Multitasking is a method for making a computer appear to be doing several things at once. In reality, it's swapping back and forth between various tasks very quickly. Nonetheless, from a human user's perspective, several things seem to be going on at the same time. You can actually do multitasking without a kernel. Consider an interrupt-driven terminal multiplexer—a device that collects lines of text from several terminals and passes them (accompanied by some sort of identifying information) along one communications channel to a single mainframe computer. You might have eight input channels, each of which would trigger a unique interrupt in the processor. There would also be eight interrupt-service routines, each of which would collect characters until an end of line was found and then send that line to the mainframe. Each routine would maintain its own input buffer and attach a unique identifier to each line. The multiplexer would appear to be doing eight things at once, reading from eight input sources simultaneously, but in reality it would be

performing eight independent tasks when stimulated by distinct external events (characters arriving). If eight characters were to arrive simultaneously, the multiplexer might fail because of insufficient time to react to all eight inputs before more characters come along.

A different approach to the same problem is to have a single interrupt, usually created by a timer chip of some sort, that starts up a scheduler subroutine. The scheduler activates a particular subroutine, just as if it had been activated by the interrupt itself. When the next timer interrupt comes along, the scheduler suspends the current subroutine and activates a new one, just as if a second, higher-priority interrupt had come along. The scheduler also takes care of some of the overhead involved with the interrupt service. In particular, it does a context swap—it saves all the registers (including the instruction pointer) and preserves the running subroutines's stack. (Each task has its own stack, just as a well-behaved interrupt-service routine should have its own stack). A task, then, is different from a normal subroutine in that it is activated only by the scheduler, not by a subroutine call in the common sense, and it has its own stack and a place to save registers during a swap.

Note that a task is not the same as a process in normal operating system parlance. A process usually implies a program that is read in from the disk and executed by the operating system. A task, however, is any independently executing code. In the case of the kernel presented here, a task is a subroutine (and the subroutines that it calls) that has its own stack and register set. That is, it is the stack and register-save area that define the task, not the code itself. Several tasks can share the same code—provided that

by Allen Holub

tween (they stay in memory until the space is needed for something else, whereupon they are overwritten, but they'll be read back into memory once space is available).

The main parts of the operating system are the kernel (or executive), the I/O system, and the shell. The

Announcing - the database development system that you designed.

TM

db_Vista III

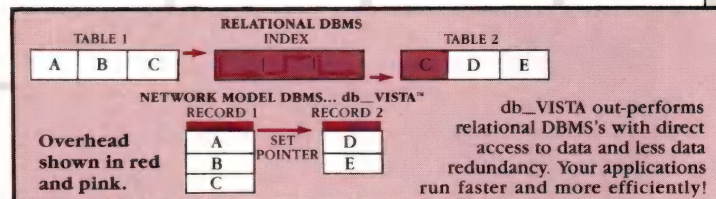
C PROGRAMMERS-

We asked what you wanted in a database development system and we built it!

db_VISTA III™ is the database development system for programmers who want powerful, high performance DBMS capabilities ... and in any environment. Based on the network database model and the B-tree indexing method, db_VISTA III gives you the most powerful and efficient system for data organization and access. From simple file management to complex database structures with millions of records, db_VISTA III runs on most computers and operating systems like MS-DOS, UNIX, VAX/VMS and OS/2. It's written in C and the complete source code is available, so your application performance and portability are guaranteed! With db_VISTA III you can build applications for single-user microcomputers to multi-user LANs, up to minis and even mainframes.

RAIMA'S COMMITMENT TO YOU: No Royalties, Source Code Availability, 60 days FREE Technical Support and our 30-day Money-Back Guarantee. Extended services available include: Application Development, Product Development, Professional Consulting, Training Classes and Extended Application Development Support.

HOW TO ORDER: Purchase only those components you need. Start out with Single-user for MS-DOS then add components, upgrade ... or purchase Multi-user with Source for the entire db_VISTA III System. It's easy... call toll-free today!



The db_VISTA III™ Database Development System

1 db_VISTA™: The High Performance DBMS

The major features include:

- Multi-user support for LANs and multi-user computers.
- Multiple database access.
- File and record locking.
- Automatic database recovery.
- Transaction processing and logging.
- Timestamping.
- Database consistency check utility.
- Fast access methods based on the network database model and B-tree indexing. Uses both direct "set" relations and B-tree indexing independently for design flexibility and performance.
- An easy-to-use interactive database access utility.
- File transfer utilities for importing/exporting ASCII text and dBASE II/III files.
- A Database Definition Language patterned after C.
- Virtual memory disk caching for fast database access.

- A runtime library of over 100 functions.
- **Operating systems:** MS-DOS, UNIX V, XENIX, VMS, OS/2.
- **C Compilers:** Lattice, Microsoft, IBM, Aztec, Computer Innovations, Turbo C, XENIX, and UNIX.
- **LAN systems:** LifeNet, NetWare, PC Network, 3Com, SCO XENIX-NET, other NET-BIOS compatible MS-DOS networks.

2 db_QUERY™: The SQL-based Query.

- Provides relational view of db_VISTA applications.
- Structured Query Language
- C linkable.
- Predefine query procedures or run ad-hoc queries "on the fly".

3 db_REVISE™: The Database Restructure Program.

- Redesign your database easily.
- Converts all existing data to revised design.

All components feature royalty-free run-time distribution, source code availability and our commitment to customer service. That's why corporations like ARCO, AT&T, Hewlett-Packard, IBM, Northwestern Mutual Life, UNISYS and others use our products.

db_VISTA III™ Database Development System

db_VISTA III™ \$595 - 3960
db_QUERY™ \$595 - 3960
db_REVISE™ \$595 - 3960

db_VISTA™ File Manager Starts at \$195

We'll answer your questions, help determine your needs and get you started.

CALL TODAY!

1-800-db-RAIMA

(that's 1-800-327-2462)



RAIMA™
CORPORATION

3055 112th Avenue N.E., Bellevue, WA 98004 (206)828-4636
Telex: 6503018237MCIUW FAX: (206)828-3131

OUR NEW GRAPHICS SOFTWARE
MAY BE A BIT FASTER THAN



ARE FOR dBASE AND R:BASE AN WHAT YOU'RE USED TO.

CURSOR	Char: Home End	Record: PgUp PgDn	DELETE	Char: Del	Insert Mode: In
Field: Home End	Page: PgUp PgDn	Field: F1	Exit: Abort	End: Esc	
Help: F1		Record: F1	Set Options: None		

TRNSID	CUSTID	EXPID	DATE	NETAMT	FGT
4768	100	131 06/11/88	27800.00	278.00	
4768	105	168 06/11/88	9500.00	95.00	
4768	104	126 06/11/88	76000.00	760.00	
4765	101	102 06/11/88	136000.00	1360.00	
4080	105	168 06/11/88	210625.00	2106.25	
4085	102	129 06/11/88	36625.00	366.25	
4778	103	131 06/11/88	152250.00	1522.50	
4775	101	102 06/12/88	87500.00	875.00	
4900	101	102 06/12/88	22500.00	225.00	
5000	101	102 06/12/88	40500.00	405.00	
5010	107	131 06/12/88	100750.00	1007.50	

DB G R S E (C:) TRNSX Dec: 3/19

View and edit fields.

DB Graphics turns raw dBASE III/III PLUS and R:BASE data files into graphics with no conversion, export or import.

It's meant plodding back and forth between your database and your graphics program, through file conversions and translations, until you finally got a chart that made your point.

But now there's a faster route to arresting presentation graphics. Now there's DB Graphics.

THE FIRST GRAPHICS SOFTWARE THAT GOES STRAIGHT TO THE HEART OF YOUR DATABASE.

DB Graphics is the only presentation graphics software designed specifically for dBASE, R:BASE and other database software users.

Since it reads dBASE and R:BASE databases directly, there are no time-consuming

export, import or conversion utilities to fool with. DB Graphics can also sort and group fields and use conditional operators and other powerful database management tools

to precisely select and graphically display the data you need. And because it has a direct line to your live data, you can modify a graph, or choose different data to represent, in an instant.

MAKE A GRAPHIC IMPROVEMENT IN ALL YOUR PRESENTATIONS.

Even though DB Graphics is more intelligent

Up until now, turning the raw data of a dBASE III/III PLUS or R:BASE file into easily understood graphs and charts has been anything but quick.

than other graphics programs, it's no less attractive. It allows you to display your data in eight different graph types—pie, bar, high/low, area, scatter, mixed, column, and line. You can work with 16 colors and hundreds of vivid combinations of textures and patterns. Plus, you can choose

from seven type styles for free text and labels.

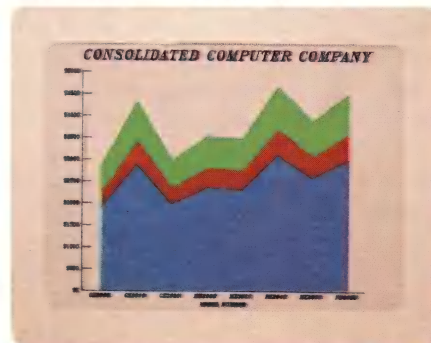
In fact, there's really only one kind of database presentation DB Graphics isn't capable of making. A dull one.

**FOR A TRIAL
PACK, CALL
1-800-624-0810
DEPT. DR1287.**

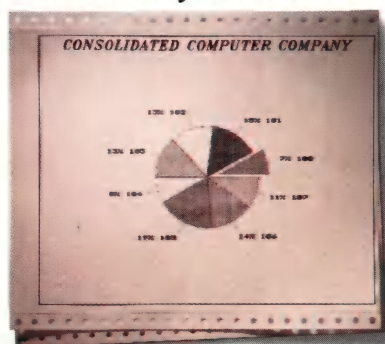
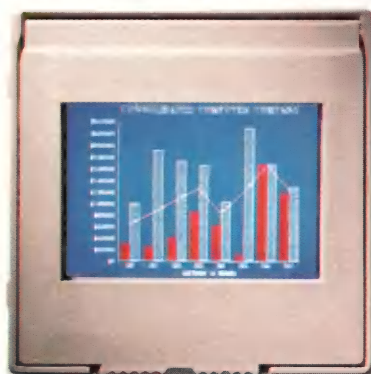
DB Graphics can produce a variety of output, including 35mm slides.

We could go on about the power, speed and accuracy of DB Graphics, but you really need to experience it for yourself.

So call our toll-free number (or from Alaska or Canada call 1-206-867-1800 Dept. DR1287) and order a DB Graphics Trial Pack for just \$9.95 today. And start making the most out of your database.



Powerful database management tools make it easy to get the data you need into a graph.



You can change the style of a graphic, or the data it represents in an instant.



The only presentation graphics software specifically designed for dBASE, R:BASE and other database software users.

DB Graphics also works with Lotus 1-2-3, Symphony, Multiplan and other data management files after translating to the ASCII, dBASE III/III PLUS or R:BASE format. DB Graphics is compatible with the IBM PS/2, PC, XT, AT and 100% compatible personal computers. It supports the IBM VGA, MCGA, EGA, CGA and Hercules display standards. It also supports most popular plotters, graphic printers, camera systems and laser printers from IBM, Epson, Hewlett-Packard and many others. © 1987 Microm.

CIRCLE 199 ON READER SERVICE CARD

C CHEST

(continued from page 126)

the code doesn't use static variables. Because each task has its own stack, and because the program counter is saved as part of the context swap, the local variables (on the stack) will be distinct and each task will remember where it was in the code when it's interrupted. Much of the context-swapping code assumes that everything is contained in a single, 8086, small-model program, however. You'll have to play with the program a little to make it support other models or external pro-

gram execution.

Several strategies are used to determine which task gets control when the timer interrupt comes along. In round-robin scheduling, the various tasks are activated one at a time until all of them have been executed once, then the processor goes around the circle again. In a priority-based system, each task is assigned a unique priority. All non-running tasks are organized into an active list. When an interrupt comes along, the scheduler compares the priority of the running task with that of the highest-priority task in the active list and, if necessary, puts

the running task to sleep and activates the top task in the list. It's possible to have a system that combines both of these strategies—tasks with the same priority are executed in round-robin fashion but a higher-priority task takes precedence over lower-priority ones.

There are two approaches to multitasking, too, that are useful in different situations. The method I've just described is preemptive multitasking. The timer interrupt causes the next task to preempt the current one. You can also have a nonpreemptive system (such as Microsoft Windows) in which individual tasks voluntarily give up control when they don't need the CPU anymore. There are two advantages to the nonpreemptive approach. You don't need a timer interrupt and CPU time is not wasted figuring out that the scheduler doesn't need to do a context swap. On the down side, a task retains control until it yields (gives up control). If the running task crashes, the whole system stops.

Because tasks aren't subroutines in the normal sense of the word, they can't pass information to each other using arguments and return values. Intertask communication is then done with a message-passing system. The operating system lets you create a set of message queues, or mailboxes, to which messages can be sent. Other tasks can then wait at the queue for a message to arrive. Several tasks can wait at the same queue—they're just given messages as they arrive, and each task gets one message. If no tasks are waiting, an incoming message is queued up until a task comes along to fetch it. A task that's waiting for a message is suspended—it will not be activated by the scheduler until a message arrives at the queue—though it's also possible for a task to time-out—the task is put back into the active list if the message hasn't arrived within a specified time. Once the message arrives, if the receiving task is of higher priority than the running task, it gets control immediately; otherwise, the task (and the message) are put back into the active list and will be reactivated in the normal way. Note that a mailbox is a data structure that contains two queues: a queue of

If You Have Turbo C You Have Half Your C-Programming Vehicle

Turbo C is a great compiler but there is one vital cog missing—debugging. Without it, you have to spend an awful lot of energy to go a short distance.

Gimpel Software's C-terp, long recognized as the leading C interpreter, now fully supports Turbo C with complete compatibility guaranteed.

Interactive Debugger—Our debugging facilities include split screen (code in upper portion, dialog in lower), breakpoints (sticky, temporary, line/function, cursor-directed), display of structures and arrays, execution of any expression (even those involving macros), function traceback with arguments, watch expressions and watch conditions (watchpoints). Our watch expressions can be structs or arrays. We catch out-of-bounds pointers!

No Toy—Full K&R with ANSI enhancements. Multiple-module with a built-in automatic make. It has virtual memory option (with optional direct use of extended memory) and a shared symbol option for those big programs. It supports graphics, dual displays and the EGA 43-line mode.

Links to external libraries—(both code and data, automatically) which can call back to interpreted functions. Function pointers are compiler compatible.

100% Turbo-C compatible.—Same header (.h) files, data alignment, bit field orderings and preprocessor variables as your compiler. We link in your compiler's library.

Our reconfigurable editor—is multifile and comes with a configuration script to mimic Turbo's editor.



The missing wheel that will turn your half-cycle into a bicycle

C-terp

Order C-terp today!

Call (215) 584-4261

Introductory Price for Turbo C-terp:
\$139.00

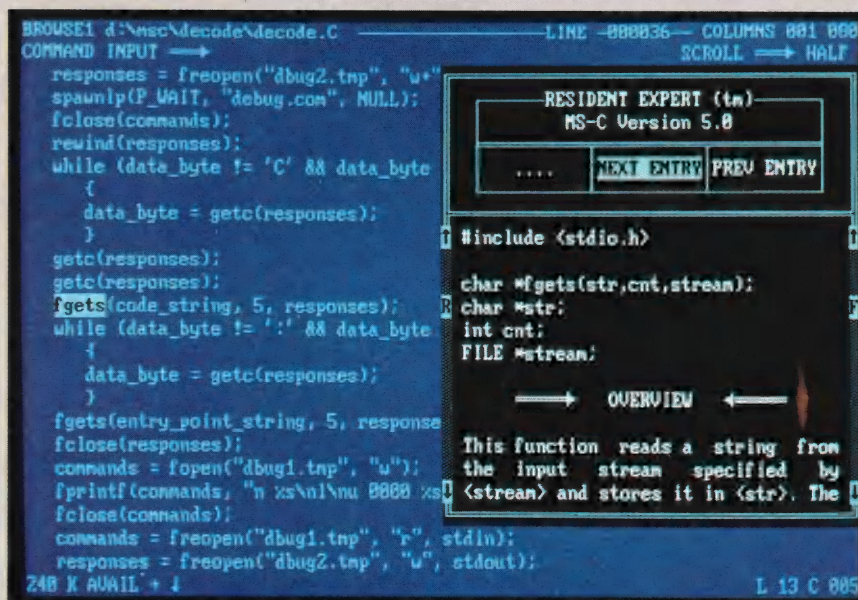
VISA, MC, COD—30 day money back guarantee

C-terp Version 3.0 is also available for the following compilers:
Microsoft, Lattice, Aztec, C86, and Mark Williams (\$298) and Xenix (\$498).



C-terp is a trademark of Gimpel Software, and Turbo C of Borland International.

RESIDENT EXPERT Pop-up Reference Guides...



Try One And Get Our MS-DOS/PC-DOS Guide Absolutely FREE!

THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name *on your screen* and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC *and* your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our *Compiler Reference Guides* contains virtually everything you need to know to program with your *preferred implementation* of your favorite language. Language syntax, all library functions, compiler directives, and error codes are thoroughly documented.

Our *PC Programmer's Reference Guide* documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

Free With Any Purchase!

Our Companion Pop-up Guide To MS-DOS 3.2/PC-DOS 3.3

Limited Time Offer

QUICK DRAW ACCESS SYSTEM

Point-and-shoot...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

Fully cross indexed...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

Multiple volumes on line...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

A COMPLETE LIBRARY...STILL ONLY A BEGINNING

At Santa Rita, our commitment is to provide the most accurate, extensive selection of PC language reference materials available. If you don't see one of our guides for your favorite language or compiler listed below don't worry, we're probably working on it!

PC Programmer's Reference Guide ... \$59.00
(with Assembly Language Guide)

Borland Turbo C (1.0) \$59.00
Borland Turbo Pascal (3.0 and below) \$59.00
(with Graphics & Numerical Methods Toolbox)

Borland Turbo Prolog (1.1 and below) \$59.00
(with Prolog Toolbox)

Lattice C Compiler (3.2 and below) \$59.00

Mark Williams LetsC (4.0 and below) \$59.00

Microsoft C Compiler (5.0 and below) \$59.00

Santa Rita

For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

Santa Rita Software
1000 E. 14th Street, Suite 365
Plano, Texas 75074

The RESIDENT EXPERT System

Resident Expert is a trademark of The Santa Rita Company. Borland, Turbo C, Turbo Pascal, and Turbo Prolog are trademarks of Borland International Inc. IBM and PC-DOS are trademarks of International Business Machines Corporation. Lattice C is a trademark of Lattice Inc. LetsC is a trademark of Mark Williams Company. Microsoft and MS-DOS are trademarks of Microsoft Corporation.

CIRCLE 201 ON READER SERVICE CARD

messages and a queue of waiting tasks.

A problem with this communications method is deadlock, a situation in which two tasks are both waiting for messages that can only be sent by the other task. That is, task A is waiting for a message that can only be sent by task B, and at the same time, task B is waiting for a message that can only be sent by task A. A similar situation can arise when every task in the system is waiting for a message simultaneously.

Another problem is resource blocking. The DOS I/O system is a good example of why this blocking is necessary. DOS is not reentrant, which means that a DOS function cannot be called reliably from an interrupt-service routine. Put another way, once you call DOS, you can't call DOS a second time until the first DOS call has finished. Because task activation can happen at any time, even when you're inside DOS, some method is needed to turn off the scheduler when a task is doing a DOS call. A task blocks to get control of the CPU—no other task will be activated as long as scheduling is blocked. Note that the system will halt if scheduling is blocked and the running task crashes.

An alternate approach to blocking that doesn't have this disadvantage is an exclusion semaphore. An exclusion semaphore is just a queue of length 1. A message is waiting at the queue when a resource (such as DOS) is available. To use the resource, a task gets the message, uses the resource, and then reposts the message to free up the resource.

This approach has its disadvantages, too. If the task that has the resource is suspended for some reason, no other task can use the resource until the task is reactivated.

Note that, though you can block by disabling interrupts, that's not usually a good idea. First of all, on the IBM PC you'll mess up the system clock, which is interrupt-driven. The other problem is the console I/O and disk subsystems, which are also interrupt-driven. You won't be able to send or receive characters when interrupts are off. You're also likely to mess up your disk transfers. If you do disable interrupts, do it for the shortest time possible. Also note that tasks must be treated as interrupt-service routines. In practice this means that you must always block before you call DOS.

Kernel Users' Manual

This section is just a users' manual for the kernel subroutines. I haven't gone into any implementation details, all of which I'll discuss next month.

Several error codes are declared in kernel.h and are returned by the various multitasking subroutines. The codes are shown in Table 1, below. `TE_NOERR` has the value 0, and the other error codes are small negative numbers. In addition, kernel.h holds `typedefs` for the TCB and `T_QUEUE`. The former is a task control block, used to hold a task's stack and so forth; the latter is a message queue. I'll discuss all of these codes in greater depth later.

Various global variables and subroutines are used internally. These are listed in Table 2, below. You shouldn't use the names in your own programs, because, for the

most part, these variables are not useful to an application program. The possible exception is `T_clock`, the system clock. `T_clock` is incremented on every system clock tick. It is not incremented while scheduling is blocked, however. Because `T_clock` is an *unsigned long*, roll-over is not a problem. If you assume the default of 18.2 ticks/second, the clock will roll over after about 65,552 hours (about 7.47 years):

$$((0xffffffff/18.2)/60)/60 = 65,552 \text{ hours} \\ 65,552/24/365.35 = 7.47798 \text{ years}$$

Of course, this number will scale with faster tick rates, but the resolution should be OK for all reasonable tick rates. The other variable of possible interest is `T_numtasks`, which holds the number of tasks that have been created so far.

There are a few additional considerations. The Microsoft compiler inserts a call to a subroutine called `_chkstk()` at the head of every subroutine. This can cause problems because a task is running on its own stack, not on the one that `_chkstk()` expects it to be using. The problem is solved by `t_start()`, which replaces the default `_chkstk()` with its own version. The new version checks the current stack pointer against the base of the task's local stack. If a stack overflow happens, multitasking is terminated and `t_start()` will return an error code (see later).

Starting and Stopping Multitasking

A typical program will first create a few tasks (I'll look at how in a moment) and then start up the multitasking environment. (Tasks can

<code>TE_NOERR</code>	No error
<code>TE_TOOMANY</code>	Maximum number of tasks (32) already exists
<code>TE_NOMEM</code>	Insufficient memory available
<code>TE_BADARG</code>	Illegal argument
<code>TE_TIMEOUT</code>	Time-out
<code>TE_QFULL</code>	Queue is full
<code>TE_NOTASKS</code>	No tasks to send message
<code>TE_INTERNAL</code>	Internal error
<code>TE_DEADLOCK</code>	Delete would have caused a deadlock
<code>TE_STACK</code>	Stack overflow
<code>TE_KILL</code>	Ctrl-Break encountered

Table 1: Error codes declared in kernel.h

TCB	*T_active;
unsigned long	T_clock;
int	T_numtasks;
T_QUEUE	*T_queues;
PQ	*T_tasks;
void	t_reschedule();
void	t_install();
void	t_swap_in();
void	t_shazam();
void	t_speedup();
void	t_slowdown();

Table 2: Global variables and subroutines used by kernel.h

DEBUGGING SWAT TEAM

*Order Eco-C88 Rel. 4.0 New Modeling Compiler
and get C-more at no extra charge!*

Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three-way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails **syntax errors** cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of **semantic error** checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is \$99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

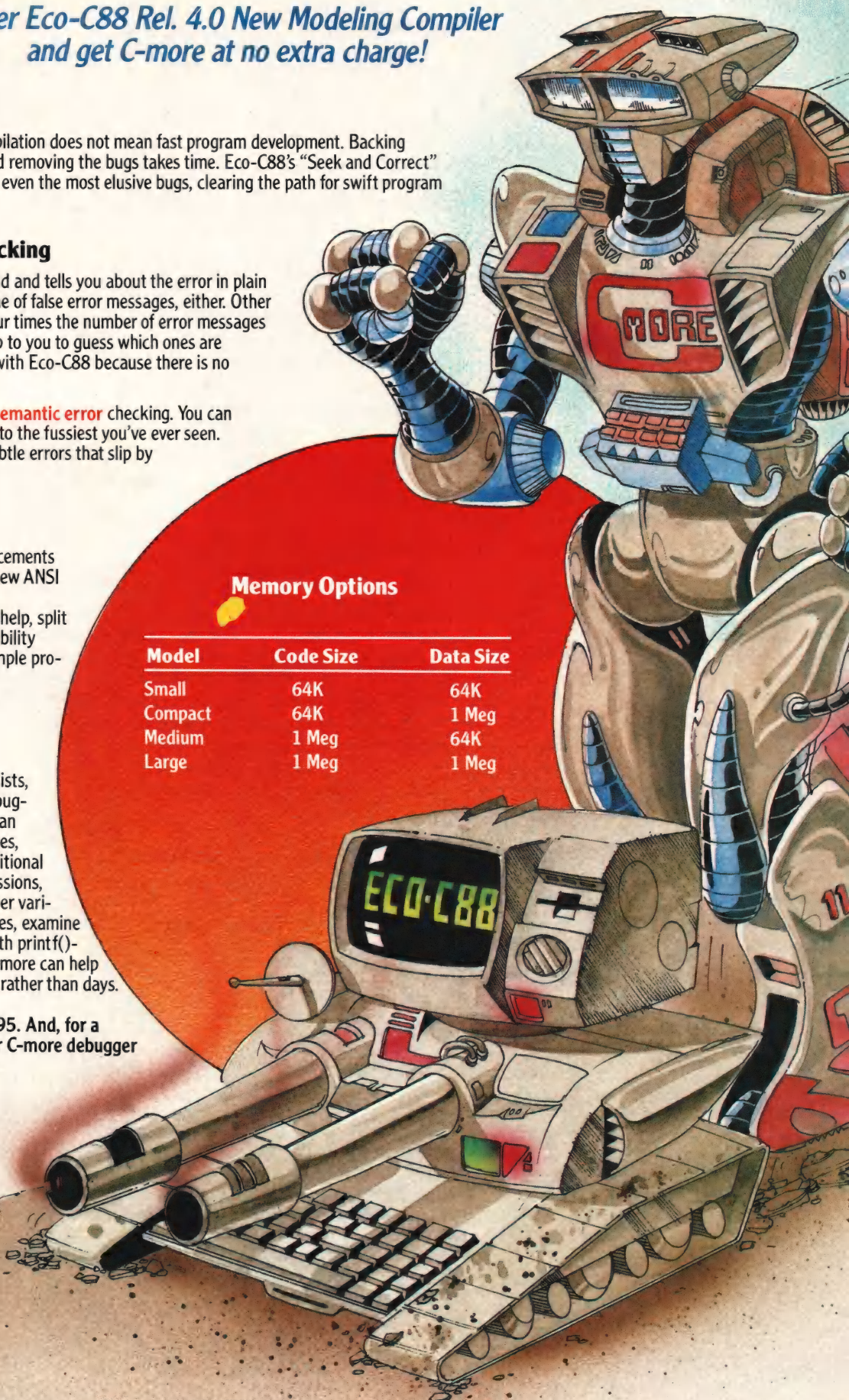
Ecosoft Inc.

6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg



create other tasks, too. They don't all have to exist before multitasking is started.) Multitasking is started with a call to:

```
int t_start( speedup )
int speedup
```

Speedup is a system-clock speedup factor. If it's 0, then the system will not be preemptive and tasks will have to yield (either with a call to *t_yield()* or by waiting for a message) to give up control of the CPU. A speedup factor of 1 uses the default PC clock rate (roughly 18.2 interrupts/second). A factor of 2 is twice as fast (36.4 interrupts/second). If the factor is too large, the system will actually slow down because it will start missing interrupts. It's best if the speedup factor is a power of 2.

At least one task must have been created prior to the *t_start()* call. Control passes immediately to the highest-priority task. *T_start()* returns either when all tasks are deleted or when *t_stop()* (discussed later) is called. *TE_NOTASKS* is returned if no tasks exist initially—multitasking is not started in this situation. *TE_NOERR* is returned when all tasks have been deleted successfully. No tasks can be waiting on queues. This is the normal way to return. *TE_STACK* is returned as soon as a stack overflow in any task is detected. *T_active* will point at the TCB of the offending task.

TE_DEADLOCK is returned when the only active task in the system deletes itself. Other tasks exist but they're all pending on queues. That

is, there must always be at least one running task in the system. To accomplish this, a very-low-priority idle task is often created. This task doesn't do anything but spin around; it's only active when all other tasks are doing something else, and it deletes itself when it's the only task left in the system. A typical idle task is shown in Example 1, below.

If *TE_NOERR* is returned, then all memory allocated to the tasks will have been restored to the heap; otherwise, if one of the above errors was returned, *T_active* will point at the TCB of the offending task. Other return values are possible if a task calls *t_stop()* directly (see later).

A panic abort from the multitasking environment can be accomplished with a call to:

```
t_stop( errcode )
int errcode;
```

Multitasking is turned off, and control passes back to the routine that called *t_start()* (immediately following the *t_start()* call). That is, *t_stop()* forces *t_start()* to return—it does not itself return. *Errcode* is passed back to the calling routine as the return value of *t_start()*. The process is analogous to a Unix *exit()* call, which doesn't return and whose argument is passed back to a *wait()* call in the parent process.

Creating and Deleting Tasks

Tasks are created with a call to:

```
TCB *t_create( subr, tag, priority,
               stack_size, ..., NULL)
int (*subr)();
```

```
char *tag;
unsigned priority;
int stack_size;
```

Subr is a pointer to the subroutine that forms the main module for the task. *Tag* is a string used to identify the TCB—it's used only for debugging. *Priority* is the task's priority—the higher the number, the higher the priority. Priorities should be in the range 0–255. If more than one task has the same priority, the tasks are executed in a round-robin fashion. *Stack_size* is the stack size (in 16-bit words) for this task only. Note that a few Microsoft functions (such as *printf()*) use up inordinate amounts of stack. If you're going to call Microsoft library routines, you'll need at least 1K stacks (*stack_size* of 512).

The remaining arguments are a *NULL*-terminated list of pointer-size arguments that will be passed to the task at start-up. (More on this in a moment.)

T_start() forces a reschedule. That is, if the task that you're creating is of higher priority than the running task, the new task will get control immediately.

A pointer to the created TCB is returned normally. This pointer is useful if you want to delete the task later. Error return values are *TE_TOOMANY* (maximum number of tasks [32] already exists) and *TE_NOMEM* (insufficient memory available to create task).

An example of task creation is shown in Example 2, page 137. Here, a single task called *foo()* is created in *main()*. It is of priority 10 and has a 512-word (1K) stack. The remaining arguments are passed to *foo()* when the task starts up. *Foo()* will print its arguments and delete itself. (It prints "hello world.")

Note that a task should never return in the normal way. It should always delete itself rather than returning. If a return is executed (either explicitly or implicitly by falling off the bottom of the subroutine), then *t_stop()* is called immediately (with a garbage argument).

A task's priority can be changed at any time with a call to:

```
int t_chg_priority( tp, new_priority )
```

```
idle(\sc128\)\n
{\n
    int t, i ;\n\n
    do {\n
        for( i = 1000; --i>=0 ; )\n            ;\n        t_cli(\sc128\);          /* Clear interrupts */\n        t = T_numtasks;          /* t = # of tasks */\n        t_sti(\sc128\);          /* restore ints. */\n    }\n    while( t > 1 ) ;\n\n    t_delete( NULL );          /* delete self */\n}
```

Example 1: A typical idle task

If you ever wanted to take a crack at assembly language, now's the time.

You probably already know that assembly language subroutines are the smartest way to get the fastest programs.

But if the complexities of working in assembler made you think twice, here's some good news. We've made Microsoft® Macro Assembler Version 5.0 a lot easier to use.

We eased the learning process by giving you the best support around. We completely revised our documentation. The new Mixed Language Programming Guide gives you step by step instructions for linking your assembly code with Microsoft QuickBASIC, C, FORTRAN, Pascal and other languages. And you get a comprehensive reference manual with listings of the instruction set and examples of each instruction. We didn't stop there, though. You also get an on-disk collection of templates and examples.

We've also dramatically simplified the high-level language interface. In just a few

simple steps, you can be calling Macro Assembler subroutines from programs written in your favorite language.

Now that you're writing the fastest programs, Microsoft is giving you the fastest way to debug them. For the first time, we've added our CodeView® debugger to Macro Assembler.

With source code and comments on your screen, Microsoft Code-

View makes debugging programs containing assembly language subroutines a snap.

And you'll be glad to know that you don't sacrifice any speed for all the ease of use.

We took the fastest Macro Assembler on the market and made it even faster.

So what are you waiting for? Get your hands on Microsoft Macro Assembler and see what it's like to break your personal speed limit.

Microsoft®

For more information or for the name of your nearest Microsoft dealer, call (800) 426-9400. In Washington State and Alaska, (206) 882-8088. In Canada, call (416) 673-7638.

Microsoft, the Microsoft logo and CodeView are registered trademarks of Microsoft Corporation.

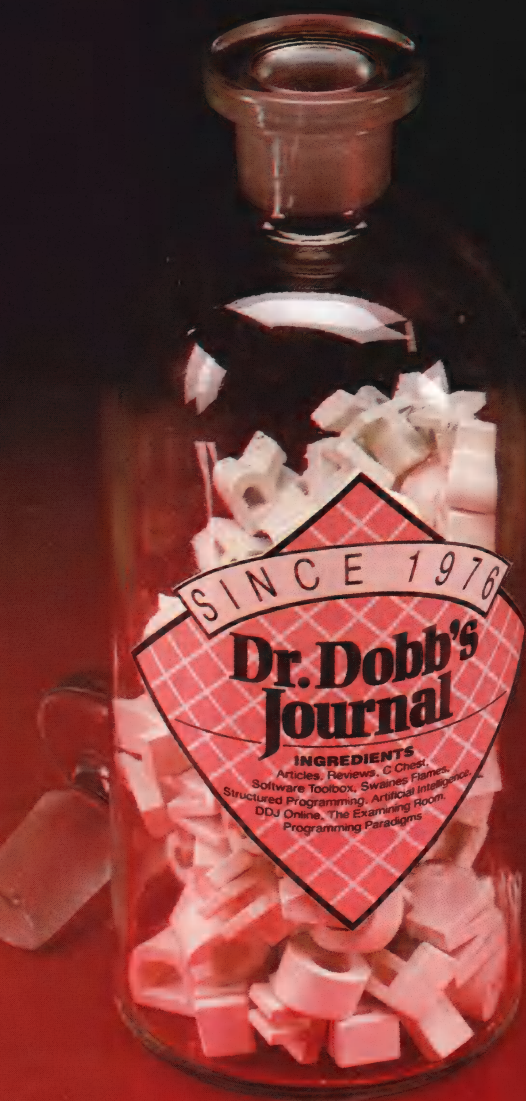
CIRCLE 203 ON READER SERVICE CARD

THE CURE FOR COMMON CODE

Are you getting the recommended monthly allowance of C, Assembly, Forth, Pascal, BASIC or Modula-2? Subscribe to *Dr. Dobb's Journal of Software Tools* and you won't catch any nasty bugs again!

Each month the Doctor brings you aid for ailing algorithms and the cure for common code. For the latest developments in software design and pages of code that will make you a more productive programmer, take the Dr. Dobb's prescription.

For more than a decade, the programming elite have known *Dr. Dobb's Journal* to be the foremost source of software tools. Subscribe now and get your monthly dose from the Doctor.



PASCAL
FOR
BASIC
MODULA-2
C
ASSEMBLY

Dr. Dobb's Journal of Software Tools

The R_x for Programmers

Subscribe
Now &

Save
Over
15%

Off the
Newsstand
Price!

SUBSCRIBE AND SAVE!

Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over \$5—a 15% savings off the cover price!

☐ 1 year \$29.97 ☐ 2 years \$56.97

☐ Please charge my:

☐ Visa

☐ Master Card

☐ American Express

☐ Payment enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

ONLY \$29.97! YOU SAVE OVER \$5.00

Savings based on a full one-year cover price of \$35.40. Canada & Mexico add \$10 for surface mail per year. All other countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3465

**\$5
SAVINGS**

SUBSCRIBE AND SAVE!

Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over \$5—a 15% savings off the cover price!

☐ 1 year \$29.97 ☐ 2 years \$56.97

☐ Please charge my:

☐ Visa

☐ Master Card

☐ American Express

☐ Payment enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

ONLY \$29.97! YOU SAVE OVER \$5.00

Savings based on a full one-year cover price of \$35.40. Canada & Mexico add \$10 for surface mail per year. All other countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3465

**\$5
SAVINGS**

COMMENTS & SUGGESTIONS

Dear Reader,

December 1987, #134

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed

Which articles or departments did you enjoy the most this month? Why?

Comments or suggestions _____

Name: _____

Address: _____



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools

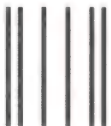
Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States



**Dr.
Dobb's
Journal
of
Software
Tools**



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States



**The
R_x
for
Programmers**

**Subscribe
Now &**

**Save
Over
15%**

**Off the
Newsstand
Price!**

PLACE
STAMP
HERE

Dr. Dobb's Journal of
Software Tools

501 Galveston Drive
Redwood City, CA 94063

C CHEST

(continued from page 134)

```
TCB *tp;
int new__priority;
```

Like `t_create()`, this routine forces a reschedule. If the task was waiting on a message, it is immediately timed-out and put back onto the active list. A task may change its own priority. This routine returns `TE_NOERR` normally and `TE_BADARG` if the task doesn't exist or if the priority of the task is greater than 255.

Tasks are deleted with a call to:

```
int t_delete( task )
TCB *task;
```

which also frees all memory associated with *task*. Note that memory that the task itself allocates (via a `malloc()` call or equivalent) is not freed—only the memory that `t_create()` allocated (for the stack and so forth) is released. A task may delete itself with a `t_delete(NULL)` call. `T_delete()` forces a reschedule when the current task is deleted. Return values are `TE_NOERR` on success and `TE_BADARG` if the task doesn't exist.

Messages

Three routines are used to create message queues (mailboxes) and for passing messages between tasks. Queues are created with a call to:

```
T_QUEUE *t_makequeue( size )
int size;
```

Size is the maximum number of messages that can be waiting in the queue. Any number of tasks can wait at a queue, however. Normally a pointer to the queue is returned, but `TE_NOMEM` is returned if there's insufficient memory. The queues are linked into a linear list that is searched for timed-out tasks on every system clock tick, so it's best to create the most active queues first.

Messages are sent from a task to a queue with a call to:

```
int t_send( q, msg )
T_QUEUE *q;
void *msg;
```

where *q* is a pointer returned from a previous `t_makequeue()` call and *msg* is a pointer to the message. The pointer, not the message, is stored, so the messages can be anything you want. Typically, *msg* will be a pointer to a structure or a string. The message itself must be in static memory.

If no tasks are waiting at the queue, `t_send()` returns immediately; otherwise, the message is attached to the task and the scheduler is called. This means that if the waiting task is of higher priority, control will be taken away from the task that sent the message and given to the waiting task. `TE_NOERR` is

returned normally. Error returns are `TE_BADARG` on a bad *q* argument and `TE_QFULL` if the queue is full.

The other side of the message-passing system is:

```
void *t_wait( q, timeout )
T_QUEUE *q;
int timeout;
```

which is used by a task to wait at a queue for a message to arrive. *Q* is a pointer returned from a previous `t_makequeue()` call and *timeout* is a time-out value. The task will only wait for *timeout* system clock ticks before it is put back onto the active list. (Remember, though, the clock

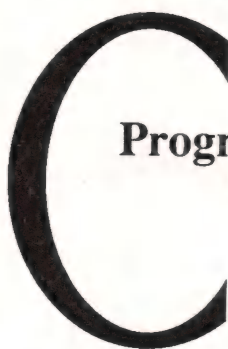
```
foo( a, b )
char *a, *b;
{
    t_printf("%s %s\n", a, b );
    t_delete( NULL );
}

main( \sc128\ )
{
    t_create( foo, "foo", 10, 512, "hello", "world", NULL );
    t_start( 2 );
}
```

Example 2: An example of task creation

'Faster and better in no time'

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.



Programmer's Toolbox Volumes I & II

- Powerful, common user interface
- Interactive and batch execution
- Online documentation
- Self-explanatory error messages

- Monitor program/system execution
- Beautify program listings
- Determine program flow/critical path
- Trace/verify variable usage
- And more....

At \$79.95 per volume or \$130 for both with a 30 day trial, the Toolbox is simply the best value today. In addition, the documentation is packed with examples and tips on creating better programs.

Why waste time, call or write us today.



MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781

won't tick when the scheduler is blocked). The maximum time-out is 32,767 clock ticks (that's about half an hour at 18.2 ticks/second). If the time-out is 0, `t_wait()` returns immediately (without a reschedule) if no messages are waiting in the queue.

Message requests are queued up in order received, without regard to priority. I've done this both because it's easy and because, in most applications, tasks with different priorities will not be pending on the same queue.

If a message is present in the queue, `t_wait()` immediately returns the pointer to the message (the same pointer as was passed into

`t_send()`) without a reschedule; otherwise, the current task is suspended (removed from the active list) until a message arrives.

Normally, a message pointer is returned. Error return values are `TE_TIMEOUT` (on a time-out or if the `timeout` argument is 0 and no message is waiting) and `TE_NO_TASKS` (if the current task is the only running task—a guaranteed deadlock).

Blocking and Yielding

Four subroutines are provided to support blocking: `void t_cli()`, `void t_sti()`, `void t_block()`, and `void t_release()`.

`T_cli()` and `t_sti()` disable and enable interrupts. Use these advisedly. Interrupts should never be off for extended periods. They should

never be off when you're using the DOS I/O functions.

`T_block()` and `t_release()` are more reliable. `T_block()` disables the scheduler but not the normal clock interrupt. That is, the current task retains control of the system once `t_block()` is called. Because interrupts are not disabled, this is a much safer routine to use than `t_cli()`. There is one caveat, however. If a normal interrupt happens while interrupts are disabled, the hardware will execute the interrupts as soon as they're enabled again. This is not the case with `t_block()`. That is, the scheduler does not know whether an interrupt happened while it was blocked. Consequently, a tight loop such as this:

```
while( *str )
{
    t_block();
    putchar( *str++ );
    t_restore();
}
```

won't work as expected. Because so little time elapses between the `t_restore()` and the next `t_block()` call, the odds are that an interrupt will never occur while the scheduler is active. That is, the following will probably work in just the same way as the previous example:

```
t_block();
while( *str )
    putchar( *str++ );
t_restore();
```

Note that a `t_cli()` can disrupt the DOS system clock (time will be lost) while interrupts are off; `t_block()` doesn't have this limitation.

The final control-related subroutine is `void t_yield()`. This routine puts the current task to sleep and activates the task that has the next-highest priority. Note that the current task is always suspended, even if it's the highest-priority task. It will get back control on the next timer interrupt in this case, however. `T_yield()` is used primarily when the scheduler is blocked or when you're running a nonpreemptive system. It's also useful when a high-priority task doesn't do much and doesn't want to hog the CPU. `T_yield` returns `TE_NOERR` when

```
#include <stdio.h>
#include <tools/video.h>
#include "kernel.h"

#define TEST 3 /* 1 = nonpreemptive test,
                * 2 = preemptive test: simple timer
                * 3 = test round-robin scheduling
                */

T_QUEUE *Queue1;
T_QUEUE *Queue2;

main(\sc128\ )
{
    int status = 0;
    long t_numint(\sc128\), t_numblk(\sc128\);
    int sam(\sc128\), dave(\sc128\), timer(\sc128\), idle(\sc128\),
                                                maintask;

    if( !(Queue1 = (T_QUEUE *) t_makequeue( 2 ) ) )
        printf("Can't make Queue1 queue\n"), exit(1);

    if( !(Queue2 = (T_QUEUE *) t_makequeue( 2 ) ) )
        printf("Can't make Queue2 queue\n"), exit(1);

    #if (TEST == 1)
        status = (int) t_create(sam, "sam", 100, 512, "SAM", NULL);
        status = (int) t_create(dave, "dave", 50, 512, "DAVE", NULL);
        status = t_start( 0 );
    #endif

    #if (TEST == 2)
        status = (int) t_create(timer, "timer", 10, 512, "timer", NULL);
        status = (int) t_create(idle, "idle", 1, 100, NULL);
        status = t_start( 2 );
    #endif

    #if (TEST == 3)
        status = (int) t_create(maintask, "maintask", 200, 512, NULL);
        status = t_start( 2 );
    #endif

    t_perror( "\ndone: ", status );
    t_sstats(\sc128\);
    printf("%ld interrupts, %ld blocked\n", t_numint(\sc128\),
                                                    t_numblk(\sc12;
}
```

Example 3: A program that creates queues, starts multitasking, and performs three tests

the yield was successful and `TE_NO_TASKS` if there were no tasks to which to give control.

Note that `t_yield()` normally doesn't return until after it regains control of the system. That is, the normal control flow goes like this:

1. A task calls `t_yield()`.
2. The second task is activated.
3. The second task is suspended.
4. Control goes back to the original task and `t_yield` returns `TE_NOERR`.

Statistics

Several debugging and statistics routines are provided. Two routines are useful for debugging:

```
t_tprint( t )
TCB *t;
```

```
t_qprint( q )
T_QUEUE *q ;
```

`T_tprint()` is passed a TCB pointer, and it prints the TCB in human-readable form to standard output; `t_qprint()` does the same for the

queues.

Several statistics routines are also useful: `long t_numint()`; returns the number of unblocked interrupts, and `long t_numblk()`; returns the number of blocked interrupts. Finally, `t_stats()` prints various scheduler-related statistics (the number of times the scheduler was called, the number of time-outs, and the number of context swaps done by the scheduler). This routine should not be called from a task (because it uses `printf()`).

Miscellany

```
int t_second()
```

Returns the number of system clock ticks in a second, given the speedup factor passed to `t_start()`. It returns 0 if the speedup factor was 0. In this case, a `t_wait()` call will never time-out.

```
char *t_errlist()
```

Works like the normal `errlist()` that's supported by most C compilers. Indexed by error code, it evaluates to a string that holds an appropriate error message.

```
t_iserr(x)
```

Returns true if `x` is an error code and false otherwise.

```
t_perror( str, errcode )
```

```
char *str;
```

Prints an error message. If `errcode` is an error code, it prints an appropriate message; otherwise, it prints `status XXX`, where `XXX` is `errcode` represented as a decimal number.

```
t_printf( fmt, ... )
```

```
char *fmt;
```

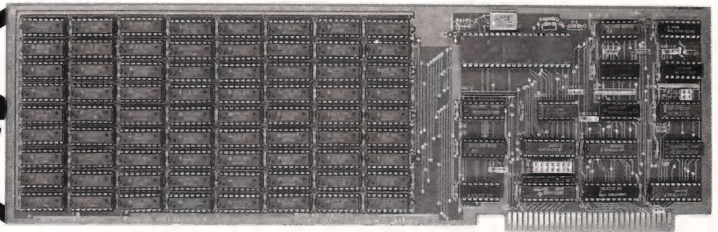
Works like `printf()` except that it blocks before printing anything.

```
T_PRIORITY(a,b)
```

```
TCB *a, *b;
```

This macro, which is in `kernel.h`, compares the priorities of the two tasks. It returns a negative number if `a`'s priority is less than `b`'s, 0 if the priorities are the same, and a positive value if task `b` is of higher priority than task `a`. Note that a task's priority takes into consideration both the priority passed to `t_create()` and the time at which the task was last suspended. That is,

SemiDisk[®] has an *attractive* personality.



"A while back I got a SemiDisk to help me with my database work. A SemiDisk is like a RAM-disk only a whole lot better. It doesn't sit in my main or EMS memory, and, using the Battery Backup, it's like permanent storage.

"That SemiDisk makes light work of the jobs that were sending my hard disk to an early grave. And SemiDisk has no head to crash; no moving parts to wear out. With all the time it saves me, I figure it paid for itself in just a couple of months.

"Then I heard programs like Microsoft Windows could use my SemiDisk for temporary files instead of using EMS. So I moved them

over to the SemiDisk, too. The quiet speed of it is almost elegant!

"My boss wanted to try my SemiDisk on the company LAN server, but I told him to get his own. A couple of days later, he was wearing a grin as big as mine. I guess he likes his SemiDisk too.

"One morning I booted up my computer and there was my word processor waiting for me on the SemiDisk. I swear I didn't put it there! After I tried it, I knew it was there to stay.

"Meanwhile, I've found a new use for my hard disk, too. It's great for backing up my SemiDisk!"

I/O mapped SemiDisk goes in standard PC, XT or AT expansion slot. Priced at just \$495 for 512K, \$795 for 2Mb. Battery Backup \$130. Up to 8Mb per drive. Call or write for further information or to place an order.

SemiDisk

End the waiting.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075
(503) 626-3104



CIRCLE 205 ON READER SERVICE CARD


```

dave( arg )
char *arg;
{
    char *s;

    t_printf( "In dave(%s), about to wait for message\n", arg );

    if( t_iserr(s = t_wait(Queue1, 100)) )
        t_perror("dave: first wait call", (int) s );
    else
        t_printf( "dave: got <%s> from Queue1\n", s );

    if( t_iserr(s = t_wait(Queue1, 100)) )
        t_perror("dave: first wait call", (int) s );
    else
        t_printf( "dave: got %s from Queue1\n", s );

    t_printf("dave: yielding\n");
    t_yield(\scl28\);

    t_printf("dave: deleting self\n");
    t_delete( NULL );
}

```

Example 4: The task sam()

```

sam( arg )
char *arg;
{
    int err;

    t_printf( "In sam(%s), yielding: no messages\n", arg );

    t_yield(\scl28\);

    t_printf( "sam: back from yield, sending messages\n" );

    if( err = t_send( Queue1, "1st message" ) )
        t_perror( "Foo: sending 1st message", err );

    if( err = t_send( Queue1, "2nd message" ) )
        t_perror( "Foo: sending 2nd message", err );

    if( err = t_send( Queue1, "3rd message" ) )
        t_perror( "Foo: sending 3rd message", err );

    t_printf("Foo: yielding again\n");
    t_yield(\scl28\);
    t_printf("Foo: Returned from 2nd yield, deleting self\n");
    t_delete( NULL );
}

```

Example 5: The task dave()

```

In sam(SAM), yielding: no messages
In dave(DAVE), about to wait for message
sam: back from yield, sending messages
dave: got <1st message> from Queue1
dave: got 2nd message from Queue1
dave: yielding
Foo: yielding again
dave: deleting self
Foo: Returned from 2nd yield, deleting self
done: No error

Scheduler called 0 times: 0 tasks timed-out, 0 context swaps
0 interrupts, 0 blocked

```

Example 6: Output from test 1 Example 3

C CHEST

(continued from page 139)

if two explicit priorities are the same, then the task that was suspended most recently is considered to be of lower priority than the earlier task. This mechanism makes round-robin scheduling easy to implement because tasks will move to the bottom of the list as they're suspended.

Some Examples

Let's look at a few examples. First, *main()* must create a couple of queues and start up multitasking. I'll use a common *main()* module for all the examples, which are shown in Example 3, page 138. One of three tests is performed, depending on the value of the *TEST* macro. Two queues, both of length 2, are created at the top of the subroutine. Then a few tasks are created and multitasking is started up with a *t_start()* call. Finally, after control returns from *t_start*, various statistics are printed.

Now let's look at the individual tests. Test 1 exercises the nonpreemptive mode and demonstrates message passing. The two tasks, *sam()* and *dave()*, are shown in Examples 4 and 5, left. They both print their start-up arguments (from the original *t_start()* call) and then pass several messages back and forth. Note that, though I'm passing strings as the messages, you could pass pointers to anything. Both the *t_wait()* calls and the *t_yield()* calls cause control to be passed between tasks. Also, *sam()* intentionally tries to enqueue too many messages to see if the error mechanism is working properly. The output from the program is shown in Example 6, left. Note that all the statistics at the bottom of the output are 0 because the system isn't preemptive.

The next test is a small timer. It uses the *idle()* task that I looked at earlier. The *timer()* task is shown in Example 7, page 141. It prints a start-up message and then enters a *for* loop that executes five times. The *t_wait()* call times-out after one second because nobody's sending any message to the queue. Consequently, the timer will print five ex-

clamation points, one every second, and then delete itself.

The third test is shown in Example 8, below. This test demonstrates several things. First of all, *main()* creates only one task, *maintask()*, which in turn creates two more tasks. Note, however, that both of these tasks use the same code! This is possible because every task has its own stack. Several tasks can share the same code, provided that they don't use static variables (the local variables will be on physically distinct stacks). Here, the task identifies itself by looking at its argument and it prints that argument every so often. The *dv_putchar()* subroutine is a direct-video output function, any of these functions will do. I generally use direct video in multi-tasking applications because going through DOS is so unpredictable (it messes with the interrupt system). Because the two tasks are of the same priority, they'll be executed in a round-robin fashion—five seconds of alternating 1s and 2s will be printed on the screen.

So that's how to use the subroutines. I'll look in depth at how they work next month.

Meaner Than Ever

Kevin Jennings writes:

"I read with some interest your column in the May 1987 issue of *DDJ* regarding the calculation of sample means. There is indeed a simple algorithm for computing the true mean of a set of data points on the fly. I've included both a block diagram [Figure 1, page 142] as well as a C function that implements it [Example 9, page 142—I've taken the liberty of making Kevin's code a little more efficient—Allen]. Call *mean()* with *reset* true the first time; thereafter, call it with *reset* false and *data* holding the current sample. The subroutine returns the running mean.

"The data comes in, one sample at a time, and is represented in the block diagram at *Z(t)*. The algorithm computes a gain to apply to the difference between this measurement and what the filter thinks the measurement should be. This gain [*K(t)*] is then added to the previous estimate to give a new estimate. If equation 2 is rearranged as:

$$\hat{x}(t) = (1-K(t))\hat{x}(t-1) + K(t)Z(t) \quad (3)$$

and you look at what *K(t)* does as the samples come in, you should be able to convince yourself that this algorithm does indeed return the

sample mean of all data received from the time that the filter is reset up to the current measurement. I don't recommend that you actually implement equation 3, however, because it's inherently less accurate

```
timer(\sc128\)  
{  
    int i;  
  
    t_printf("Starting up timer\n");  
  
    for( i = 5; --i >= 0 ; )  
    {  
        t_printf( "!" );  
        t_wait( Queue2, t_second(\sc128\ ) );  
    }  
  
    t_printf("Deleting timer\n");  
    t_delete( NULL );  
}  
#endif
```

Example 7: The timer() task

```
timer( arg )  
{  
    /* I'm assuming that pointers & ints are  
     * the same size here.  
     */  
  
    int i;  
  
    while( 1 )  
    {  
        for( i = 10000; --i >= 0 ; )  
        ;  
        t_cli(\sc128\);  
        dv_putchar( arg + '0' );  
        t_sti(\sc128\);  
    }  
}  
  
maintask(\sc128\)  
{  
    TCB *t1, *t2;  
  
    t1 = t_create( timer, "1st timer", 100, 512, (void*)1, NULL);  
    t2 = t_create( timer, "2nd timer", 100, 512, (void*)2, NULL);  
  
    t_perror( "task1:", (int)t1 );  
    t_perror( "task2:", (int)t2 );  
  
    t_wait( Queue2, 5 * t_second(\sc128\ ) );  
  
    t_delete( t1 );  
    t_delete( t2 );  
    t_delete( NULL );  
}
```

Example 8: Test 3, used in Example 3

than is 2. Statisticians will recognize this algorithm as a least-squares estimator. It's also called a Kalman filter in digital signal processing.

"Note that it's more convenient to update the reciprocal of the gain $[1/K(t)]$ than to update $K(t)$ directly. Because the reciprocal takes on only positive integer values, it's tempting to declare ki as *unsigned int*, but then ki would wrap to 0 when it reached the maximum value for *unsigned int* (65,535, given a 16-bit *int*). You'd have to correct for the wrap-around by holding ki at its maximum value once it's reached. Thereafter, the algorithm would no longer give you the exact mean, but no

other algorithm would either. You're just trying to process too much data. I snuck around this difficulty by declaring ki as *double*. (*Doubles* typically don't wrap when they overflow). In any event, because you have to do a floating-point divide to calculate the mean, there's no advantage in storing ki in an *unsigned int* because the compiler would have to convert it to *double* to do the arithmetic."

Kevin doesn't point out that the algorithm as implemented requires several floating-point operations, so it's slower than the exponential smoothing algorithm I presented in May, which used nothing but shifts and addition. On the other hand, the Kalman filter has an appealing simplicity and is more appropriate

than exponential smoothing in many applications. It's certainly more accurate for small numbers of samples. Don't be tempted, by the way, to change the *doubles* into *floats*. Most C compilers convert all *floats* to *doubles* whenever they're used in an expression. Consequently, it takes longer to multiply two *floats* than it does to multiply two *doubles* because you have to convert them to *doubles*, do the arithmetic, and then truncate back down to *floats* to store the result. Moreover, any space savings that you get by using *floats* are usually lost in the additional code needed to do the type conversion.

Availability

All the source code for the multi-tasking kernel described both this and next month (including the priority-queue stuff) is available for \$30 on an IBM-PC 5¹/₄" disk from Software Engineering Consultants, P.O. Box 5679, Berkeley, CA 94705. Include local sales tax if you're ordering from California. In addition to the kernel code and the priority queue routines, the disk includes an enhanced version of the curses window I/O package described in the July 1987 C Chest. Because the enhanced curses uses direct video reads and writes rather than going through DOS, it's useful in multi-tasking applications that can't use the DOS I/O functions. This version of curses supports overlapping windows (though you can only write to the top one) and lets you delete and move windows. In addition, it lets you create boxed windows (Unix's curses doesn't).

DDJ

(Listings begin on page 110.)

Vote for your favorite feature/article.
Circle Reader Service No. 6.

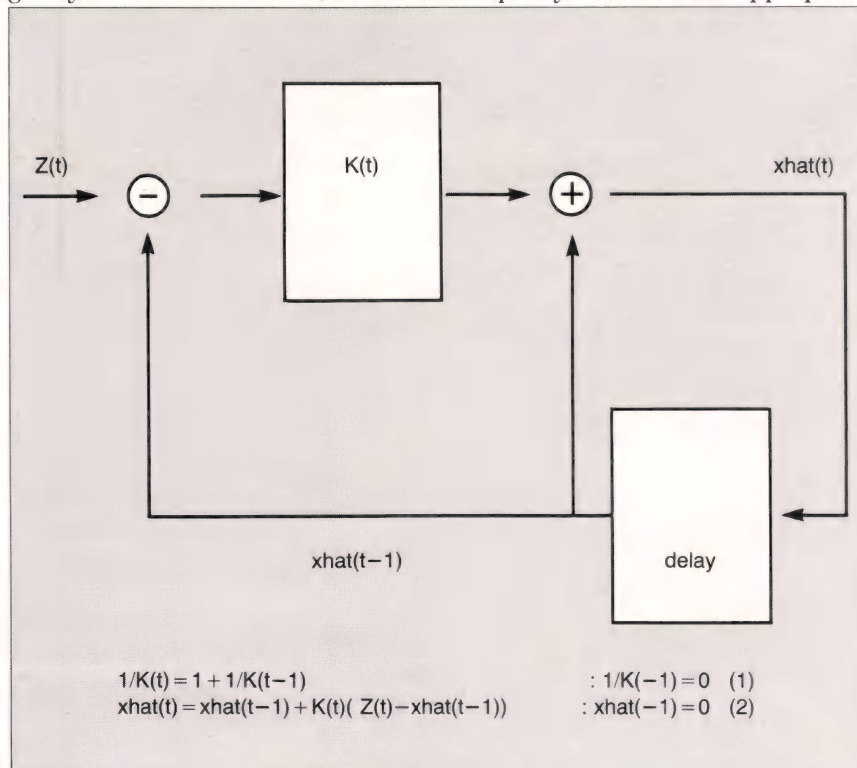


Figure 1: Block diagram of an algorithm that computes the true mean of a set of points on the fly

```
double mean(reset, data)
int reset;
double data;
{
    static double xhat, ki;

    return reset ? (ki = xhat = 0)
        : (xhat += (data - xhat) / ++ki)
        ;
}
```

Example 9: A C function that implements the algorithm shown in Figure 1

C CODE FOR THE PC

source code, of course

C Source Code

FSP (screen manager)	\$400
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	\$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$275
Vitamin C (MacWindows)	\$200
Essential C Utility Library (400 useful C functions)	\$160
Essential Communications Library (C functions for RS-232-based communication systems)	\$160
Panache C Program Generator (screen-based database management programs)	\$150
PC/IP (CMU/MIT TCP/IP implementation for PCs)	\$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$100
The Profiler (program execution profile tool)	\$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	\$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	\$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	\$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	\$80
ME (programmer's editor with C-like macro language by Magma Software)	\$75
Wendin PCNX Operating System Shell	\$75
Wendin PCVMS Operating System Shell	\$75
Wendin Operating System Construction Kit	\$75
EZ.ASM (assembly language macros bridging C and MASM)	\$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	\$50
Make (macros, all languages, built-in rules)	\$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	\$50
Coder's Prolog (inference engine for use with C programs)	\$45
PC/MPX (light-weight process manager; includes preemption and coroutine packages)	\$45
Biggerstaff's System Tools (multi-tasking window manager kit)	\$40
TELE Kernel (Ken Berry's multi-tasking kernel)	\$30
TELE Windows (Ken Berry's window package)	\$30
Clisp (Lisp interpreter with extensive internals documentation)	\$30
Translate Rules to C (YACC-like function generator for rule-based systems)	\$30
6-Pack of Editors (six public domain editors for use, study & hacking)	\$30
ICON (string and list processing language, Version 6 and update)	\$25
LEX (lexical analyzer generator)	\$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	\$25
C Compiler Torture Test (checks a C compiler against K & R)	\$20
PKG (task-to-task protocol package)	\$20
A68 (68000 cross-assembler)	\$20
Small-C (C subset compiler for 8080 and 8088)	\$20
tiny-c (C subset interpreter including the tiny-c shell)	\$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	\$20
List-Pac (C functions for lists, stacks, and queues)	\$20
XLT Macro Processor (general purpose text translator)	\$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	\$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	\$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	\$60
Webster's Second Dictionary (234,932 words)	\$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TeX or bitmap format)	\$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	\$15
U. S. Map (15,701 points of state boundaries)	\$15

The Austin Code Works

11100 Leafwood Lane

Austin, Texas USA 78750-3409

(512) 258-0785

FidoNet 382/12

Free surface shipping on prepaid orders

MasterCard/VISA

CIRCLE 206 ON READER SERVICE CARD

New Forth Sources, a Bibliography, and String Extensions for Forth-83

In the news this month is FORCE, Harris Semiconductor's version of the Novix NC4016 Forth processor. FORCE stands for Forth-optimized RISC computing engine. Chuck Moore's original hardware design has been copied into the Harris cell library, where it can be rapidly moved to various semiconductor technologies. Harris has fixed the NC4016 bugs and has added some extra features, such as byte addressability. FORCE is intended to be the smart heart of custom VLSI chips for knotty problems with high-speed solutions.

According to Dave Williams of Harris, a FORCE-based, real-time control processor (RTCP) will be available in the first quarter of 1988. The RTCP chip will contain an interrupt processor, 256-word data and return stacks, and a 16×16 hardware multiply. (Rumor has it that it will be offered on an IBM PC card shortly thereafter.) Dave says the chip will run at 15 MHz, or more than 15 million instructions per second. FORTH Inc. intends to support the chip with a variation of the same polyFORTH it developed for the Novix NC4016. This Forth includes an optimizing compiler that can pack sequential instructions together so that they run simultaneously. It also includes extensive fixed, fractional, and floating-point math support and a flexible nonlinear curve fitter (the mathematical equivalent of a monkey wrench).

Speaking of Forth chips, Dr. C. H.

by Martin Tracy

Ting's *More on NC4000, Volume 5*, is now available. This newsletter contains 80 pages of technical nitty gritty on the Novix NC4016 (originally numbered the NC4000). Volume 5 contains a reprint of "The FORCE Toolbox" by Dave Williams of Harris Semiconductor. You can



order *More on NC4000* for \$15 from Offete Enterprises Inc. at (415) 574-8250. Back issues are still available.

Also available from Dr. Ting at Offete is the *F83 Reference Manual* (1987). This is strictly a reference manual and is meant to accompany the popular public-domain Laxen/Perry F83 Forth. Words are arranged by topic, and each word has a one-line description. A total of 400 words are described in 40 pages, which include an index and a complete catalog of Offete's other publications—all this for only \$10.

What's surprising is not that 400 words are included but that more than 600 words are left out! (The Forth-83 Standard only requires about 200 words.) To quote Dr. Ting: "The only problem with F83, like any good looking and hard working wife, is that it is too wordy."

GENie Forth Forum

The GENie (General Electric Network for Information Exchange) Forth Forum is now in operation. This new Forth-oriented bulletin board is sponsored by the Forth Interest Group (FIG). GENie is reportedly the largest electronic information exchange network, with local-access phone numbers from most major cities.

Alan Furman writes: "Use 1,200 bps, even parity, 7 bits, 1 stop bit, half-duplex (echo on). Dial up the sign-on modem line: (800) 638-8369. It helps to record the session on disk as there is a lot of scrolling off. After CONNECT, type HHH (without CR) or just wait 5-10 seconds. The U# = prompt will come on. At the

prompt, type XJM11849,GENIE (and a CR). When the menu comes on, get the local node and billing information before signing up."

Signing up with this number results in the FIG member's discount deal: first three hours free (but regular \$18 sign-up fee). The easiest way to sign up is by credit card number. GENie will make a confirming phone call a few days later. When they do, have a 3- to 12-character moniker ready (no embedded spaces). You can change your name later, but it will cost you \$10. The \$18 initial fee includes a nicely packaged users' manual that will arrive in about a week.

Once you are on GENie, type FORTH at the command line to go directly to the Forth conference. The sysops are Dennis Rufer (D.RUFFER), Scott Squires (S.W.SQUIRES), and Gary Smith (GARY-S). Use the ATT command to see who else has "attended." The categories are:

1. FIG Bulletin Board
2. FIG Real Time Conference
3. FIG Software Library
4. About the Roundtable
5. Roundtable News

You can download a copy of Laxen/Perry F83 Forth or one of the many files contributed by Gerald Shifrin of the East Coast Forth Board. Some categories are open to FIG members only.

GENie is definitely a non-prime-time activity at \$5/hour; prime time costs \$35/hour. User assistance is available until 9:00 PM Pacific Time at (800) 638-9636.

Forth Bibliography

The third edition of *A Bibliography of Forth References* (1987) is now available from the Institute for Applied Forth Research, P.O. Box 27686, Rochester, NY 14627. This 2,000-entry bibliography references arti-

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db_VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and horizontal scrolling. Clean screen fields per screen limited only by development. String fields. Easy to painting. Bind as much data as de data entry with commas. Ask a programming library. Hexadecimal or No fields. Float fields. Quick C. Speaker functions. Lattice. Create Slug. Numeric validation routines. keystroke level. Customize screens 30 day money back guarantee. Gen assortment of editing commands. windows. Assign validation data to credentials. Pull down menus. Sup mode. All functions are kept in C style function reference. Pop-up functions. Numeric range checking. tive function names. Date and time Capture screens from existing as deep as desired. Easy to main checking. Date and time conver definition language based on C's ly definable borders. The current cally highlighted. Create reports.

Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View windows. Read only fields. Rich assortment of editing commands. Pass- Includes ROM BIOS driver. Fields can support any data type. Scrolling/ included. Specify writeable and non-writeable positions within fields.

machine. Number of memory. Fast screen modify. Fast screen sired to fields. Numeric bout our linear pro fields. Long fields. Yes Read only fields. reports. Codename Validate data at the and menus at run time. eric data pointer. Rich Easy to learn. Pop-up fields. Corporate C ports EGA 43 line separate modules. Full prompt and message No royalties. Descrip- conversion routines. programs. Nest screens tain. Run time error sion functions. Screen printf. Time fields. Ful- field can be automati- Exploding borders.

C-scape 2.0

with

Look & Feel™

The state-of-the-art interface management system preferred by professional C programmers and consultants worldwide.

Look & Feel

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

C-scape 2.0

- Windows, windows, windows
- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

Oakland Group, Inc.

675 Massachusetts Avenue
Cambridge, MA 02139-3309

800-233-3733
617-491-7311

CALL
NOW



PC/MS-DOS \$279, plus shipping (includes C-scape, Look & Feel, source, manual and support). UNIX/others call. 30-day review.

readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Turn Dan Bricklin slides into C. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Aztec, Lattice, Microsoft, UNIX and others. And that's not all. Call for demo.

CIRCLE 207 ON READER SERVICE CARD

THE FORTH COLUMN (continued from page 144)

cles and books from Forth's dim origins up to January 1987. Articles are indexed both by subject and author.

Because many Forth programmers love to write, the author index reads like a *Who's Who* of Forth. I use the subject index mostly, though. It's a great help in convincing software managers or potential customers that Forth has already been used successfully in their line of work.

Here is an example of using the index:

Q: What's a QUAN?

A: (433) The Quan Concept Expanded
(1120) Code Field Vectoring
(1538) High Speed, Low Memory Consumption Structures

Q: Has Forth been used for radar antenna programming?

A: (359) The Development of a Com-

puterized Antenna Range Field
(879) Microprocessor Control of Automated Antenna Ranges
(1010) Ground Control Approach Radar Performance Monitoring

The bibliography was edited by Thea Martin and was generated entirely with Forth programs provided by Dick and Jill Miller of Miller Microcomputer Services (MMS). Thea writes:

"We used MMS' DATAHANDLER-PLUS and FORTHWRITE to produce this document. DATAHANDLER-PLUS is a powerful, flexible database written in Forth that interfaces to the MMS word processor, FORTHWRITE. All three editions of the Bibliography have been produced with these systems."

You can order a copy of the bibliography for \$25 directly from the Institute, or you can get it from FIG ([408] 277-0668).

ANS Forth Meeting

The first meeting of ANSI X3J14 (ANS Forth) was held on August 3 and 4 at CBEMA headquarters in Washing-

ton, D.C. In attendance were:

Greg Bailey, Athena Programming (Novix Forth)
Gary Betts, Saba Technology
Ronald D. Braithwaite, The Tools Group
Richard Burton, National Bureau of Standards
Don Colburn, Creative Solutions Inc. (MacFORTH)
Chris Colburn, Creative Solutions Inc. (MacFORTH)
Ted Dickens, The Dickens Co.
John Dorband, NASA GSFC
Ray Duncan, Laboratory Microsystems Inc. (PC/FORTH, UR/FORTH)
Douglas Fishman, National Bureau of Standards
Lawrence P. Forsley, Laboratory for Laser Energetics (Rochester Conference, JFAR)
Charlie Keane, PPI
Guy M. Kelly (IEEE representative, chairman of the FST)
Charles H. Moore, Computer Cowboys (inventor of Forth)
Mike Nemeth, Computer Sciences Corp. (MD FIG, Goddard FUG)
David C. Petty, Digital

€ PROGRAMMERS! THE TOOLS YOU NEED AT A PRICE YOU'LL LIKE

BTree

Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported.

75.00

ISAM

Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language.

40.00

lp

Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used.

75.00

Snake

The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system.

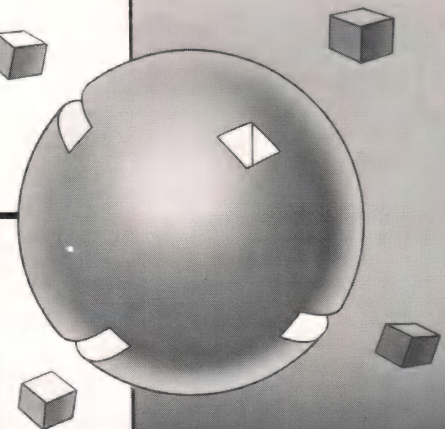
59.00

Combine & Save: BTree + ISAM + lp **159.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

**NOW
MULTI-
USER
AVAILABLE
60.00**



softfocus

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 208 ON READER SERVICE CARD

James Rash, NASA GSFC
 Elizabeth D. Rather, FORTH Inc.
 (polyFORTH)
 Gerald A. Shifrin, MCI Telecommuni-
 cations (ECFB sysop)
 Bill Ragsdale, Dorado Systems (foun-
 der of FIG, figForth)
 Robert Smith, Maxtor (FST secretary)
 Martin Tracy, FORTH Inc. (Master-
 Forth, DDJ)

The editorial comments are claims to fame and not necessarily current agenda. Ms. Rather served as the acting chair and Mr. Duncan as the acting secretary. Ms. Cathie Kachurik offered welcome assistance to the X3J14 Technical Committee (TC) on behalf of CBEMA.

The complete unofficial minutes of this meeting are available on the MCI ANS Forth bulletin board, with a copy on the East Coast Forth Board ([703] 442-8695). (Minutes become official when approved at the next meeting.) Unofficially, here are some of the highlights:

- Ms. Rather observed that a Forth standard should identify and docu-

ment accepted practice and not be an instrument to advance the state of the art. She hoped that an ANS Forth would be a clear and complete statement of what Forth is and does, that it would require minimal changes to existing systems, that it would be universally accepted, and that it would lead to the acceptance of the Forth language as a professional instrument.

- The Forth-83 Standard, Chapters 1-12, without the appendices, was approved as the Basis Document. (The Basis Document is successively modified until it becomes the Draft Proposal.) All members of the TC will receive a Basis Document with all paragraphs numbered for their comments and review.

- A proposal to add floating-point math to the Scope of Work was defeated.

- Volunteers were solicited for officer positions. The ANSI SMC selects officers from this pool of volunteers. Ad hoc committees were created to research existing practice and identify major areas of noncompliance to Forth-83. (The Research Commit-

tee has since mailed a questionnaire to all identified producers of Forth.)

The next ANS Forth meeting is scheduled for November 11 and 12, just prior to the Forth Convention. By the time you read this, both will be over. To be a voting member on the ANS Forth TC, you must pay CBEMA a \$200 fee and be prepared to attend four three-day meetings each year. You can follow the progress of the dpANS (draft proposal, pronounced "de-pants") on the MCI ANS Forth bulletin board, GENie, or through this column. For now, any technical proposals should be sent to the acting secretary (me) at FORTH Inc., 111 N. Sepulveda Blvd., Manhattan Beach, CA 90266.

Some Controlled Words

In the October 1987 issue of DDJ, I presented a Forth-83 software prelude for writing portable source code. You will need to add the words presented in that article to your Forth-83 Forth to enjoy the fruits of this column. Example 1, page 148, contains typical defini-

Parallel Programming for "C"

INTERWORK

A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	Operating System	Price
IBM PC, XT, AT	PC-DOS 2.0 or later	\$129
IBM PC AT	XENIX*	\$159
DEC VAX; SUN III	UNIX 4.2BSD	\$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.

Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add \$2.50. Send check or money order to:



Block Island Technologies
 Innovative Computer Software

13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892
 (503) 241-8971

*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

CIRCLE 210 ON READER SERVICE CARD

Conquer Time and Space.

Introducing XO-SHELL™ Pop-Up Productivity for Programmers.

No matter what language you program in, XO-SHELL will help you hurdle the barriers to working faster and more efficiently by eliminating programming hassles. Only with RAM-resident XO-SHELL can you:

- DO CROSS-REFERENCING without leaving your editor
- VIEW ANY FILE and TRANSFER ANY SECTION into your editor or to your printer
- VIEW, COPY and ERASE files directly from a SCROLLABLE DIRECTORY DISPLAY
- With a single-key stroke RETRIEVE previous DOS commands, then EDIT and REEXECUTE them
- DO SOURCE-LISTING while in your application
- OBTAIN KEY-CODES without a reference and without going through difficult interpretation
- INSERT GRAPHICS CHARACTERS in your source code.

XO-SHELL is for PCs, XT's, AT's, PS/2's, compatibles.

Call now for special half-price introductory offer.



WYTE CORPORATION
 701 Concord Avenue
 Cambridge, MA 02138

For a limited time only

\$24.50

(regular price \$49)
 plus \$5 shipping & handling

Call today toll-free

(800) 635-5011

(In MA: (617) 868-7704)
 Visa, MasterCard

CIRCLE 209 ON READER SERVICE CARD

THE FORTH COLUMN

(continued from page 147)

tions, which you can adjust to fit your Forth.

Now we can define a (small) number of additional words that have come into general usage:

```
: \ >IN @ 64 + -64 AND >IN ! ;
      IMMEDIATE
( "backslash" comment to the end
  of line.)
: THRU ( n n2) 1+ SWAP DO I LOAD
      LOOP ;
\ LOAD blocks n through n2.
```

These words are defined early on so that we can use them to compile other useful words. The "backslash" comment has two cousins:

```
: \ 1024 >IN ! ; IMMEDIATE
\ comment to end of source block.
: \IF ( f ) 0= IF [COMPILE] \ THEN ;
\ conditional comment or interpret
  line.

IMMEDIATE
```

The first word, `\`, stops interpreting or compiling a block immediately. Most Forths can use `EXIT` for this

function. The second word, `\IF`, is both a convenient comment and a key for conditional execution. I'll discuss how you can use `\IF` in a moment.

In the meantime, consider another useful word:

```
: 2* ( n - n' ) DUP + ;
```

Many programmers assume this word is required by the Forth-83 Standard. (It isn't, but `2/` is.) By defining `2*` as high-level Forth, we guarantee that we can use it in any Standard program. But chances are excellent that `2*` is already in our dictionary as a `CODE` definition. If we redefine it, the new `2*` will be much slower than the original. What we would really like to do is to define `2*` only if it isn't already in the dictionary:

```
: NEED ( - f )
\true if the following word is already
\in the dictionary.
  32 ( ie blank) WORD FIND SWAP
  DROP 0= ;
```

Now we can have our cake and eat it too:

```
NEED 2* \IF : 2* ( n - n' ) DUP + ;
```

`NEED` looks `2*` up in the dictionary. If it isn't there, we define it in high-level.

There are a few caveats to this approach. `\IF` can only be used within a source block. `NEED` returns either of two truth values, 1 or -1, so be careful if you use it in logical expressions. Finally, we assume that if `2*` is already in the dictionary, its function is to double a number and not to, say, print two stars (**) on the terminal.

Fortunately, the Forth-83 Standard has provided for this eventuality by including `2*` in its Controlled Reference Words with the appropriate definition. Controlled Reference Words are not required by the Standard, but if they do appear, they must have the prescribed definition. Example 2, below, contains some other Controlled Reference Words.

Unfortunately, there are not very many words in the Controlled Reference Set. (The Standard also has Uncontrolled Reference Words, which are exactly that.) Of course, we can define or redefine any word we want to. This is one of Forth's

```
( Moves NEXT address to and from stack.)
: I> ( - a ) COMPILE R> ; IMMEDIATE
: >I ( a ) COMPILE >R ; IMMEDIATE

( Even address alignment, if required.)
: ALIGN HERE 1 AND ALLOT ;
: REALIGN ( a - a' ) DUP 1 AND + ;

( Hides number of bytes per word.)
2 CONSTANT CELL
: CELL+ ( n - n' ) 2+ ;
: CELLS ( n - n' ) 2* ;

( Compiles self-reference.)
: RECURSE ( ... ) ; IMMEDIATE

( Forces interpretation of the input stream.)
: INTERPRET ( ... ) ;

( Discards return stack overhead of DO--LOOP.)
: UNDO I> R> R> 2DROP >I ;
```

Example 1

```
NEED D2* \IF : D2* ( d - d' ) 2DUP D+ ;
NEED HEX \IF : HEX ( DECIMAL ) 16 BASE ! ;

NEED C, \IF : C, ( n ) HERE 1 ALLOT C! ;
NEED BL \IF 32 CONSTANT BL ( a blank)

NEED ERASE \IF : ERASE ( a n ) 00 FILL ;
NEED BLANK \IF : BLANK ( a n ) BL FILL ;

NEED .R \IF : .R ( n w ) >R DUP 0< R> D.R ;
```

Example 2

```
: 2>R ( n n2)
\ pushes a pair on the return stack.
  COMPILE SWAP COMPILE >R COMPILE >R ;
  IMMEDIATE

: 2R> ( - n n2)
\ pops a pair from the return stack.
  COMPILE R> COMPILE R> COMPILE SWAP ;
  IMMEDIATE

: @EXECUTE ( ? ) @ EXECUTE ;

: AGAIN
\ used in a BEGIN-- AGAIN structure.
  0 [COMPILE] LITERAL [COMPILE] UNTIL ;
  IMMEDIATE

: DLITERAL SWAP
  [COMPILE] LITERAL [COMPILE] LITERAL ;
  IMMEDIATE

: S>D ( n - d ) DUP 0< ;
\ single to double number.

: WITHIN ( n min max - f )
\ true if min <= n < max.
  OVER - >R - R> U< ;

-1 CONSTANT TRUE
```

Example 3

greatest benefits—nothing is sacred (except Forth itself). Anytime we redefine a *CODE* definition, however, we lose speed needlessly.

The solution is to add a few selected high-performance words to the Controlled Reference Set. Because the Forth-83 Standard committee is not meeting at this time, we will have to do it ourselves. So, consider the words in Example 3, page 148, as belonging to the *DDJ* Forth Column Controlled Reference Word Set. These definitions should each be preceded by the appropriate *NEED* phrase. Any other definitions we require, we will simply define (or redefine) in high-level Forth. In other words, we won't be needing *NEED* anymore.

Strings

This month's topic of interest is implementing strings in Forth. Let's see how we can apply our new tools to this issue.

The most enthusiastic string package I have ever seen was written by George Hawkins and is available as the file *FSTRINGS.ARC* from the

ECFB. George's package provides 66 definitions on 50 screens of source code, including a test suite. There are 21 pages of documentation with a glossary. The package is written in Forth-83 but uses the *BRANCH* experimental extension. If your Forth doesn't *BRANCH*, change the following definitions on screen 6:

```
: $, HERE OVER @ 2+ DUP ALLOT
      ALIGN CMOVE ;
: ($LIT) I> DUP
      DUP CELL+ SWAP @ + REALIGN
      >I ;
: ($LIT) COMPILE ($LIT) $, ; IMMEDIATE
```

Once I made this change, it compiled the first time and added a modest 3.3K to my dictionary.

Although the *FSTRINGS* package is not a *SNOBOL*, it is suitable for medium-strength string processing, such as building concordance tables. We can approximate many of its functions using materials that lie readily at hand.

First, we need some way to make strings. In Forth-83, strings are

stored in memory as counted strings. A counted string is referred to by a single address, which points to the count byte of the string. The count byte is followed by that many 1-byte ASCII characters. In some Forths, it may be padded with a blank or zero to the next even address. You can think of a counted string as a packed string that must be unpacked to a text string before it can be used. Text strings are referred to by two arguments: the address of the first character and the length of the string, with the length on top. Counted strings are converted to text strings with the *COUNT* operator. We will also need an operator to pack text strings into counted strings:

```
: PLACE ( a1 n a2) 2DUP ! 1+ SWAP
      CMOVE ;
\ packs string a1 n into counted
\ string a2.
: , 34 ( ie ASCII ") WORD COUNT
\ compiles following string at HERE
  DUP >R HERE PLACE R> 1+
      ALLOT ALIGN ;
CREATE TEST , " This is a test."
```

TURBO PASCAL USERS

AZATAR DOS TOOLKIT

NEW

No Other Product Like It - The Azatar DOS Toolkit is a collection of Pascal routines that may be used in programs to access the more powerful and more useful DOS 3.X interrupts and functions.

Substantial Power and Control Using our routines, you gain extensive power over files, directories, drives, the system's clock and switches, the DOS PRINT spooler, auxiliary devices, the keyboard and screen, and more.

Easy to Use - Just call one of our procedures or functions from within your program, and pass it simple parameters.

Complete Error Handling - In the event that an error does occur, numeric codes and extensive English messages immediately tell you which error it was, where it occurred, why it occurred, and what you can do to correct it.

Educational Examples - The Azatar DOS Toolkit clearly demonstrates how the routines may be used in actual programming contexts using sample Pascal programs.

Thorough Reference Manual - The Azatar DOS Toolkit Reference Manual contains over 225 pages of helpful introductions and overviews, explanations of all global variables used in the routines, lists of the error messages, and a great deal of other important information.

Inexpensive - The entire Azatar DOS Toolkit package is modestly priced at \$95.00. That's less than the price of the IBM DOS Technical Reference Manual alone.

Worth Every Penny - The routines in the Azatar DOS Toolkit will save programmers hundreds of hours of valuable coding time.

Call Dennis Hunter
(716) 385-9780



MICROSYSTEMS INC.

3300 Monroe Ave., Rochester, NY 14618

CIRCLE 211 ON READER SERVICE CARD

THE FORTH COLUMN

(continued from page 149)

TEST COUNT TYPE This is a test. ok

We can compile strings as literals into colon definitions in this way:

```
: (") I> COUNT 2DUP + REALIGN
>I ;
: " COMPILER (") ; IMMEDIATE
\ fundamental string compiler.
: TEST " This is a literal." TYPE ;
TEST This is a literal. ok
```

I have included *COUNT* in the run-time action of a string literal. Several Forths include the " operator but not all of them *COUNT* the string. Nonetheless, because we are redefining it ourselves, it will behave as we expect.

We can also make a string by *EXPECTING* it from the user into *PAD*:

```
( For example:)
: TEXT ( - a n)
PAD 80 EXPECT PAD SPAN @ ;
: NAME?
CR ." Sign in please:" TEXT
CR ." Your name is : " TYPE ;
```

Substrings are trivial in Forth:

```
: /STRING ( a n n2 - a' n')
\ shorten string a n by n2 characters.
ROT OVER + ROT ROT - ;
: TEST " Catatonic" ;
TEST 6 - TYPE Cat ok
TEST 4 /STRING TYPE tonic ok
TEST 5 /STRING 2- TYPE on ok
```

We can make the " literal *STATE*-smart to make testing easier. While we're at it, let's have a *STATE*-smart ASCII, too:

```
: ASCII ( - c) BL WORD 1+ C@
\ value of following character.
STATE @ IF [COMPILE] LITERAL
THEN ;
IMMEDIATE
: (") I> COUNT 2DUP + >I ;
: " ( - a n) STATE @
IF COMPILE (") ;
ELSE ASCII " WORD COUNT >R
PAD I CMOVE PAD R> THEN ;
IMMEDIATE
ASCII A EMIT A ok
" Simplicity" TYPE Simplicity ok
```

Numbers are easily made into strings with the flexible "sharp" op-

erators: <#, #, #s, *HOLD*, *SIGN*, and #>. Making strings into numbers, however, is a little harder. We'll use a syntax suggested by Stephen Pelc ("Proposed Standard Changes," *FORML Conference*, 1986):

```
VARIABLE DPL
: VAL? ( a n - d 2 , n 1 , 0)
\ string to number conversion.
PAD OVER - SWAP OVER >R
CMOVE
BL PAD C! PAD DPL ! 0 0 R>
DUP C@ ASCII - = DUP >R - 1-
BEGIN CONVERT DUP C@ DUP
ASCII : =
SWAP ASCII , ASCII / 1+ WITHIN
OR
WHILE DUP DPL ! REPEAT
R> SWAP >R IF DNEGATE THEN
PAD 1- DPL @ - DPL ! R> PAD =
IF DPL @ 0> IF DROP 1 ELSE 2
THEN
ELSE 2DROP 0 THEN ;
```

When the smoke clears, there will be a 2 on the stack if the number is a double, a 1 if it is single, and a 0 if it is invalid. The number, if any, will be on the stack under the flag. A number containing punctuation from the set +,-/: is a double number, with *DPL* set to the number of places to the right of the rightmost punctuation. A number without punctuation is a single number, and *DPL* is set to a negative value.

```
" 123,456" VAL? . 2 ok
D. 123456 ok
DPL @ . 3 ok
```

Many Forths already support automatic single- and double-number conversion, but there is no general agreement on the syntax. The Forth-83 *DPL* is one of the Uncontrolled Reference Words, which means we can't count on it. Hopefully, this issue will be taken up by the ANS Forth technical committee.

Because strings are often converted to double numbers, irrespective of punctuation, we could use a simpler conversion primitive based on *VAL?*:

```
: VAL ( a n - d f) VAL? 3 < AND DUP
\ string to double number.
\ True if number is valid.
IF 1 = IF S>D THEN TRUE EXIT
THEN
```

Brand New From Peter Norton A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot*
features programmers need

only
\$50

Direct from the man who gave you *The Norton Utilities*, *Inside the IBM PC*, and the *Peter Norton Programmer's Guide*.

THE NORTON EDITOR



"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities*. You can *program your way to glory* with *The Norton Editor*."

Peter Norton



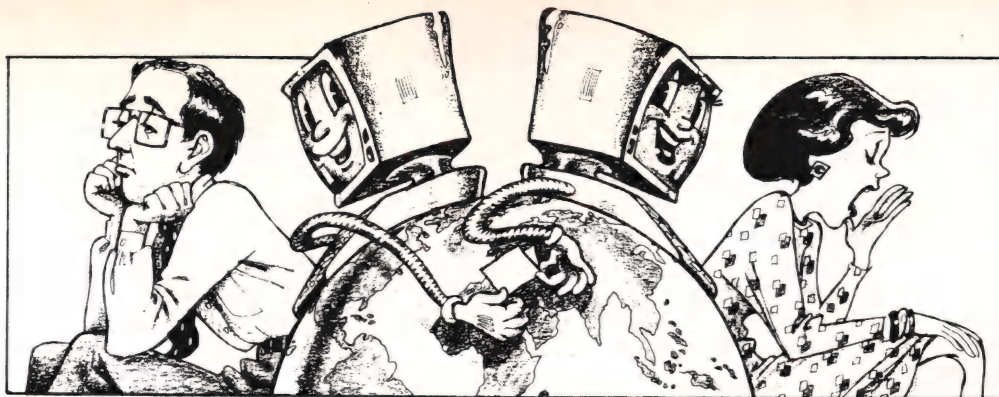
*Easily customized, and saved
Split-screen editing*

*A wonderful condensed/outline display
Great for assembler, Pascal and C*

Peter Norton Computing, Inc., 2210 Wilshire Boulevard,
Santa Monica, CA 90403, 213-453-2361. Visa,
Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing

CIRCLE 212 ON READER SERVICE CARD



Ten Reasons Not to Use PeaceNet

1. I don't like working with others

PeaceNet is a computer network and communication system for people who believe that global planning and cooperation are necessary to reverse a trillion-dollar-per-year arms race; it is linking users throughout the United States and in over 70 other countries.

2. I've got all the information I'll ever need

PeaceNet is for those who appreciate that information is always growing and changing; its bulletin boards, conferences, and databases provide information about everything from Central America to Star Wars.

3. I love playing phone tag

PeaceNet's electronic mail system renders those endless conversations with secretaries and answering machines obsolete.

4. I don't know how to use my computer

PeaceNet helps novices with simple, entertaining manuals and round-the-clock staff for answering their questions.

5. I enjoy copying, labeling, and stamping letters

PeaceNet enables you to send messages to hundreds of other users with one simple command.

6. I've got plenty of money to waste on postage and phone bills

PeaceNet is for people who want to save money; it lets you send documents across the world faster than Federal Express™ for pennies per page.

7. I don't mind getting action alerts a week late

PeaceNet does mind and can help your organization send out time-urgent alerts instantly.

8. I don't have the right kind of computer equipment

PeaceNet is available to anyone with a computer terminal and a modem.

9. An effective peace movement isn't worth 50 cents a day

PeaceNet users disagree.

10. It's all hopeless, anyway

Then why read this magazine when Modern Wrestling would suffice?

Use PeaceNet's electronic mail, conferences, bulletin boards, databases, and Telex services to gather, organize and share valuable up-to-date and comprehensive information about:

Peace and anti-nuclear efforts, Central American issues, environmental protection, anti-apartheid efforts, human rights, Native American issues, and much more.

PeaceNet is the largest and most rapidly growing on-line community of progressives anywhere in the world. And by using any home computer and a modem it is only a local phone call away. Call or write today for more information.



PeaceNet:

3228 Sacramento Street
San Francisco, CA 94115 (415) 923-0900


```
0 0 ;
```

SKIP and *SCAN* are two important string search primitives that appear in several Forths as factors of the word *WORD*:

```
SKIP ( a n c - a' n')
```

```
-TEXT ( a n a2 - -1 , 0 , 1)
\ -1 if string a n < a2 n , 0 if equal,
\ and 1 if >.
```

```
COMPARE ( a n a2 n2 - -1 , 0 , 1)
\ -1 if string a n < a2 n2 , 0 if equal,
\ and 1 if >.
```

```
-MATCH ( a n a2 n2 - offset 0 , ? -1)
\ position of string a2 n2 in a n.
\ Offset is 0 if a n is found in 1st
\ position. True with invalid offset
\ if a2 n2 isn't in a n.
```

```
: ANIMAL " ANIMAL" ;
```

```
" ANIMATE" ANIMAL COMPARE . 1 ok
```

```
" ANT" ANIMAL COMPARE . -1 ok
ANIMAL " IMA" -MATCH . . 0 2 ok
ANIMAL " XYZ" -MATCH . . -1 ????? ok
```

```
\ shortens a string to the first position
```

```
\ unequal to c.
```

```
SCAN ( a n c - a' n')
```

```
\ shortens a string to the first position
```

```
\ equal to c.
```

SKIP and *SCAN* mimic the 8086 *SCAS* instruction. Their high-level definitions are given in the accompanying source screens

(see Listing One, page 124). *SCAN* is especially useful for lexing and parsing strings:

(For example:)

```
: LEX ( a n c - a2 n2 a3 n3)
```

```
\ splits string at c, right most string
```

```
\ on top. Either string can have 0 length.
```

```
>R 2DUP R> SCAN
```

```
ROT OVER -
```

```
ROT ROT DUP 0>
```

```
NEGATE /STRING ;
```

```
" FORTH.COM" ASCII .
```

```
LEX
```

```
2SWAP TYPE SPACE
```

TYPE FORTH COM ok

CTO ("C-to-quote") is an often neglected primitive for converting a character to a string:

```
: CTO ( c - a 1) CTEMP C! CTEMP 1 ;
```

```
ASCII A CTO TYPE A ok
```

The remaining string operators (shown in Example 4, left) are used to compare strings and to find the occurrence of one string in another. Their analogs are often found in line and screen editors.

The final string operator, *EVAL*, is also the most powerful. *EVAL* interprets a string. The following definition of *EVAL* should work on your Forth-83 system. I say "should" because there is some question as to whether the Standard words *>IN*, *BLK*, *TIB*, and *#TIB* control interpretation or simply describe it. You will need to test this word on your system.

```
: EVAL ( a n ) \ interprets the string.
DUP >R TIB SWAP CMOVE R@
#TIB !
0 >IN ! 0 BLK ! INTERPRET R>
>IN ! ;
```

```
: TEST " LATER" EVAL ;
: LATER . " A forward reference." ;
TEST A forward reference. ok
```

This completes the string package, and you will find a complete listing of it in Listing One. It compiles to about 1K of dictionary space.

Availability

Most of the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listing begins on page 124.)

Vote for your favorite feature/article.
Circle Reader Service No. 7.

Example 4

The Advanced
Programmer's Editor
That Doesn't Waste Your Time

For DOS, Microport
UNIX, or ...

OS/2
Protected Mode

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Multiple windows, files
- Unlimited file size, line length
- 30 day money-back guarantee
- Great on-line help system
- Regular Expression search
- Supports large displays
- Not copy protected

Only \$195

Lugaru
Software Ltd.

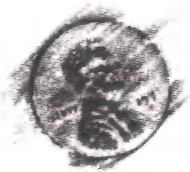
5740 Darlington Road
Pittsburgh, PA 15217

Call
(412) 421-5911

for IBM PC/XT/AT's or compatibles



Remember,

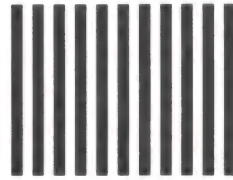


**Smart Buying
Decisions**



Start with DDJ

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

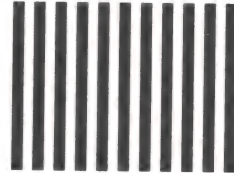
FIRST CLASS PERMIT #200, DALTON, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Dept.
P.O. Box 507
Dalton, Mass. 01227

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT #200, DALTON, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Dept.
P.O. Box 507
Dalton, Mass. 01227

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____
 Title _____
 Company _____ Phone (_____) _____
 Address _____
 City/State/Zip _____

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

December '87: Use before March 31, 1988

1. Did you buy this issue on a newsstand? () Yes () No
2. Are you a subscriber? () Yes () No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal*? () Yes () No

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____
 Title _____
 Company _____ Phone (_____) _____
 Address _____
 City/State/Zip _____

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

December '87: Use before March 31, 1988

1. Did you buy this issue on a newsstand? () Yes () No
2. Are you a subscriber? () Yes () No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal*? () Yes () No

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

For Free Info ...

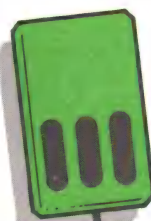
Start Here



Smart buyers start with *DDJ's* free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ's* free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!



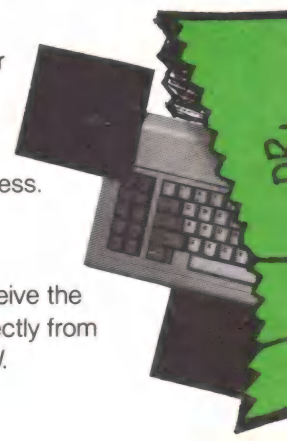
Take a Reader Service Card with You

It's Easy as ...

1. Circle the appropriate free information numbers, referring to the advertiser index for more information.

2. Fill in your name and address.

3. Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



The Advertiser Index

Advertiser Name	Page #	RS#	Advertiser Name	Page #	RS#
Addison Wesley	119	195	Oasys	111	186
AI Architects	112	187	Oregon Software	83	156
Aker Corporation	44	124	Peacock Systems	99	178
Aldebaran Laboratories	47	126	Periscope Co. Inc.	28	113
Apex Software Corporation	109	185	Pharlap	99	176
ASI/American Software International	79	150	PIM Publications	117	194
Aspen Scientific	37	120	Polytron	15	108
AT&T Electronic Photography & Imaging	10-11	106	Production Language Corp.	92	166
Austin Code Works	143	206	Programmer's Connection	157-159	217
Azatar Computer Systems, Inc.	149	211	Programmer's Paradise	66	141
Blaise Computing	2	102	Programmer's Power Pack	76	*
Block Island Tech.	147	210	Programmer's Shop (The)	97	172-174
Borland International	1	101	Programmer's Shop (The)	96	171
Bryte Computer	83	155	QCAD Systems, Inc.	50	128
Burton Systems Software	90	163	Quantum Software	72	145
C Users Group	90	162	Quarterdeck Office Supplies	13	107
Centurion Discount Software	156	215	Raima Corporation	127	*
CNS, Inc.	92	167	Roundhill Computer Systems LTD	77	149
Coder's Source (The)	82	154	SAS Institute	89	*
Coder's Source (The)	56	133	Santa Rita Software	131	201
Cogent Software Ltd.	79	151	Scantel Systems Limited	63	139
Compaq Computer Corporation	32A	*	Scientific Endeavors	87	160
Comptech	42	122	Secom Information Products Co.	99	177
Compu View	25	111	Seidl Computer Engineering	61	137
Computer Technology Group	94	169	Semaphore, Inc.	113	190
Creative Programming	93	*	Semi-Disk Systems	139	205
Crosstalk Communications	C6	219	Sharpe Systems Corporation	155	214
DDJ Subscriptions	136	*	SLR Systems	52	131
Desktop A.I.	94	170	Softfocus	146	208
Digitalink	70-71	144	SoftScience Corporation	43	123
Disk Software	115	191	Software Connections Inc.	80	152
Ecosoft, Inc.	133	202	Software Garden Inc.	67	142
Essential Software	C5	218	Software Link	21-23	110
Fair-Com	41	121	Software Security, Inc.	36	119
GenSoft Systems	81	153	Solution Systems	29	114
Gimpel Software	130	*	Solution Systems	49	127
Golden Bow Systems	103	180	Stony Brook Software, Inc.	121	196
Greenleaf Software	27	112	Tangent Designs, Inc.	113	188
Guidelines Software	45	125	Tenon Software	63	*
Hauppauge Computer Works	60	136	Texas Instruments	16-17	*
IMSI	57	134	Tool Makers (The)	156	216
Intel Corporation	58-59	135	TSF (The Software Family)	35	118
JYACC	53	132	Turbo Power Software	34	117
Kadac Products Ltd.	117	193	Turbo Tech Report	95	*
Komputerwerk	113	189	Wallsoft	125	197
Laboratory Microsystems, Inc.	73	146	Whitewater Group (The)	65	140
Lattice, Inc.	69	143	Wordcraft	87	159
Logic Process Corporation	115	192	Wyte Corporation	147	209
Logitech, Inc.	9	105	Xenosoft	90	164
Lugaru Software Ltd.	152	213			
M Street Software	50	129			
M&T Catalog of Books & Software Tools	104A	*			
Machine Independent Software Corp.	105	181			
Magma Systems	91	165			
Manx Software Systems	7	104			
Meridian Software Systems	51	130			
Metagraphics Software Corporation	101	179			
MetaWare Incorporated	85	158			
Micro Way	75	148			
Microprocessors Unlimited	84	157			
Microrim	128-129	199			
Microsoft	135	203			
MMC AD Systems	137	204			
Mortice Kern Systems, Inc.	32	115			
MSJ Subscriptions	40	*			
Nanosoft Associates	74	147			
Norton Utilities (The)	150	212			
Norton Utilities (The)	4-5	103			
Nostradamus	107	182			
NWP Intelligent Solutions, Inc.	33	116			
Oakland Group, Inc. (The)	145	207			

*The advertiser prefers to be contacted by phone; consult ad.

Advertising Sales Offices

Midwest

Charles Shively (415) 366-3600

Northern California/Northwest

Lisa Boudreau (415) 366-3600

Northeast

Cynthia Zuck (718) 499-9333

Martha Brandt (415) 366-3600

Southern California/AZ/NM/TX

Michael Wiener (415) 366-3600

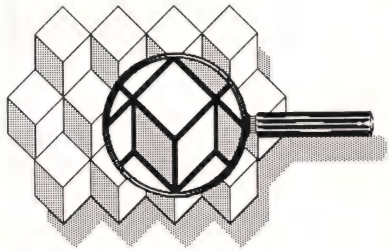
Director of Marketing and Advertising

Ferris Ferdon (415) 366-3600

Telemarketing Representative/SE/SW USA

Cheri Blum (714) 761-0294

OF INTEREST



Operating Systems

VenturCom has announced Venix System V 2.3, a real-time Unix operating system for the IBM PC/AT, HP Vectra Plus, and Compaq Deskpro 386. Enhancements include improved memory management to minimize I/O; a more efficiently coded kernel; a larger, 1,024-byte file system; and an extended buffer cache.

Venix 2.3 also features real-time extensions, adherence to AT&T standards, EGA support, AT&T binary compatibility, developers' tools that include a large-model C compiler, and Ethernet TCP/IP support.

The full system is priced at \$990 for single-quantity purchases but is being offered at an introductory price of \$600 for a two-user system until December 31, 1987. It comes with an indexed, four-volume set of documentation and 60 days of toll-free telephone support. Reader Service No. 16.

VenturCom Inc.
215 First St.
Cambridge, MA 02142
(617) 661-1230

PC-MOS/386, Version 1.02, from **The Software Link**, now supports IBM PCs; IBM PC/ATs; and compatible 8088-, 8086-, and 80286-based systems. PC-MOS/386 offers file sharing, user security, print spooling, disk caching, NETBIOS emulation, EMS emulation, interrupt-driven serial I/O, enhanced command-line recall and editing, support for very large disk volumes, an enhanced directory structure, sophisticated command processing, a full-screen text editor, and on-line help. Features

that are specific to the 80386 CPU, such as support for 32-bit native-mode applications, are now isolated in a single system driver file.

PC-MOS/386 is available on both 5¹/₄- and 3.5-inch disks. Prices range from \$195 for a 1-user version to \$995 for a 25-user version. Registered users of earlier versions can upgrade to Version 1.02 at no charge. Reader Service No. 17.

The Software Link
3577 Parkway Ln.
Atlanta, GA 30092
(404) 448-5465

Digital Research is now shipping enhanced versions of Concurrent DOS 386 and Concurrent DOS XM (Expanded Memory).

Concurrent DOS 386 supports most PC-DOS/MS-DOS applications and more than 700 business applications written specifically for Concurrent DOS. Windowing capabilities are provided, allowing up to four applications to run from the primary console. It is available in a three-user system configuration, priced at \$395, or in a ten-user version for \$495.

Concurrent DOS XM offers improved EGA support, enhanced support for the IBM PC/AT keyboard, disk formatting for up to four partitions per hard disk, and support for the AST-Four Port/DOS card and the four-port Hostess Multiport Network Adapter. The three-user system is available for \$295 and the six-user version for \$395. Reader Service No. 18.

Digital Research
P.O. Box DRI
Monterey, CA 93942
(408) 649-3896

The Hyperspace Z-System for HD64180-compatible and Z280-compatible microprocessors is now available from **Echelon**. The Hyperspace Z-System is a CP/M 2.2-compatible operating system that includes ZCPR 3.3, ZRDOS 2.0, and a sample BIOS for 64180 machines. The system features a large free memory area of 57.25K. The retail price is \$195. Reader Service No. 19.

Echelon Inc.

885 N. San Antonio Rd.
Los Altos, CA 94022
(415) 948-3820

UniPress Software has launched the Unix Training Center, which offers 22 courses to help users and organizations learn to use Unix. The center personnel can provide needs analyses, curriculum design, original course development, and instructor training. The Unix Training Center's classes can be given either at an organization's facility or at an off-site meeting area determined by a client. Classes are offered on-site for a fee of \$100 per day per student, with a minimum fee of \$1,300 per day. Reader Service No. 20.

UniPress Software
2025 Lincoln Hwy.
Edison, NJ 08817
(201) 985-8000

The Wendin-DOS Application Developer's Kit is now available from **Wendin**. The kit allows application programmers to develop multitasking applications based on VAX/VMS system services. The new QIO (Queued I/O) and RMS (Record Management System) services replace antiquated MS-DOS calls and induce more programming flexibility. Other services support shared memory for interprocess communication, memory-resident pipes, semaphores, file locking, file permissions, extended memory access, swapping, and control of multiple terminals. The kit sells for \$99. Reader Service No. 21.

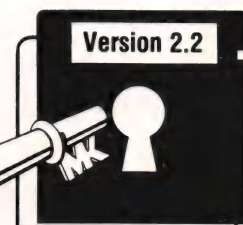
Wendin Inc.
P.O. Box 3888
Spokane, WA 99220-3888
(509) 624-8088

Microprocessor Engineering has released its OS-9 implementation of the MPE/Nautilus Cross Compiler. MPE/Nautilus features interactive debugging that allows development of the target system in RAM before committing to ROM. Written in Forth, MPE/Nautilus includes automatic handling of defining words and support for forward referencing. Reader Service No. 22.

Microprocessor Engineering Ltd.

MASTER*KEY

Unlocks Everything!



turn this
into this!

```
C:\>DEBUG PROGRAM.COM
-D100 136
8848:0100 EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20 k.Incorrect DOS
8848:0110 76 65 72 73 69 6F 6E OD-0A 24 50 B4 30 CD 21 86 version..#P40H!
8848:0120 E0 3D 36 01 72 05 3D OA-02 76 09 BA 02 01 B4 09 '=6.r...v...4.
8848:0130 CD 21 CD 20 58 EB 2F M:M Xk/
-G
```

MASTER*KEY No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!

MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!

YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 4.0 compatible).

MASTER*KEY - Easy To Use!

MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:

256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

```

H00100: JMP      Short H0011A                      ;00100 EB18      --
;-----
      DB      "Incorrect DOS version"              ;00102 49E636F727265
      DB      ODh                                  ;00117
      DB      0Ah                                  ;00118
      DB      "e"                                  ;00119 24
;-----
H0011A: PUSH     AX                                ;0011A 50          P
      MOV     AH,30h                               ;0011B B430      _O
      INT     21h                                  ;0011D CD21      _!
      XCHG    AH,AL                                ;0011F 86E0
      CMP     AX,0136h                             ;00121 3D3601    =6_
      JB      H0012B                               ;00124 7205      z_
      CMP     AX,020Ah                             ;00126 3D0A02    =_
      JBE     H00134                               ;00129 7609      v_
H0012B: MOV     DX,0102h                           ;0012B BA0201    ---
      MOV     AH,09h                               ;0012E B409      --
      INT     21h                                  ;00130 CD21      _!
      INT     20h                                  ;00132 CD20      -
;-----
H00134: POP      AX                                ;00134 58        X
      JMP     Short H00166                          ;00135 EB2F      _/
;-----

```

Page 1

MASTER*KEY XREF - PROGRAM.XRF

0102h	:	121	2F5	301	320
020Ah	:	126			
03CBh	:	12B			
1-Display_String	:	130	591	610	
1-DOS_Ver_Number	:	11D			
H00100	:	100			
H0011A	:	100	11A		
H0012B	:	124	12B		
H00134	:	129	134		
H00166	:	135			
TERM_normally:20h	:	132			

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept. 44, La Verne, CA 91750 714-596-0070

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

CIRCLE 214 ON READER SERVICE CARD

Order Now!
Just \$79⁹⁵

Phone orders accepted on MC or VISA
\$82.45 (includes \$2.50 shipping)
\$87.65 in California (includes tax & shipping) C.O.D. orders add \$2.00
(714) 596-0070

Please send MASTER*KEY!

Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept. 44, La Verne, CA 91750

Name _____
Address _____
City _____ State _____ Zip _____

You Do The Hard Part

You Do This...

We'll Do This...

FILE MAINTENANCE
1. Pat [strange]
2. Pat [strange]
3. Pat [strange]
4. Pat [strange]
5. Pat [strange]
6. Pat [strange]
7. Pat [strange]
8. Pat [strange]
9. Pat [strange]
10. Pat [strange]
11. Pat [strange]
12. Pat [strange]
13. Pat [strange]
14. Pat [strange]
15. Pat [strange]
16. Pat [strange]
17. Pat [strange]
18. Pat [strange]
19. Pat [strange]
20. Pat [strange]
21. Pat [strange]
22. Pat [strange]
23. Pat [strange]
24. Pat [strange]
25. Pat [strange]
26. Pat [strange]
27. Pat [strange]
28. Pat [strange]
29. Pat [strange]
30. Pat [strange]
31. Pat [strange]
32. Pat [strange]
33. Pat [strange]
34. Pat [strange]
35. Pat [strange]
36. Pat [strange]
37. Pat [strange]
38. Pat [strange]
39. Pat [strange]
40. Pat [strange]
41. Pat [strange]
42. Pat [strange]
43. Pat [strange]
44. Pat [strange]
45. Pat [strange]
46. Pat [strange]
47. Pat [strange]
48. Pat [strange]
49. Pat [strange]
50. Pat [strange]
51. Pat [strange]
52. Pat [strange]
53. Pat [strange]
54. Pat [strange]
55. Pat [strange]
56. Pat [strange]
57. Pat [strange]
58. Pat [strange]
59. Pat [strange]
60. Pat [strange]
61. Pat [strange]
62. Pat [strange]
63. Pat [strange]
64. Pat [strange]
65. Pat [strange]
66. Pat [strange]
67. Pat [strange]
68. Pat [strange]
69. Pat [strange]
70. Pat [strange]
71. Pat [strange]
72. Pat [strange]
73. Pat [strange]
74. Pat [strange]
75. Pat [strange]
76. Pat [strange]
77. Pat [strange]
78. Pat [strange]
79. Pat [strange]
80. Pat [strange]
81. Pat [strange]
82. Pat [strange]
83. Pat [strange]
84. Pat [strange]
85. Pat [strange]
86. Pat [strange]
87. Pat [strange]
88. Pat [strange]
89. Pat [strange]
90. Pat [strange]
91. Pat [strange]
92. Pat [strange]
93. Pat [strange]
94. Pat [strange]
95. Pat [strange]
96. Pat [strange]
97. Pat [strange]
98. Pat [strange]
99. Pat [strange]
100. Pat [strange]

SCREEN CODE (PRINT, SUBROUTINE)
9000 LOCATE 1, 1 PRINT
9001 LOCATE 2, 1 PRINT
9002 LOCATE 3, 1 PRINT
9003 LOCATE 4, 1 PRINT
9004 LOCATE 5, 1 PRINT
9005 LOCATE 6, 1 PRINT
9006 LOCATE 7, 1 PRINT
9007 LOCATE 8, 1 PRINT
9008 LOCATE 9, 1 PRINT
9009 LOCATE 10, 1 PRINT
9010 LOCATE 11, 1 PRINT
9011 LOCATE 12, 1 PRINT
9012 LOCATE 13, 1 PRINT
9013 LOCATE 14, 1 PRINT
9014 LOCATE 15, 1 PRINT
9015 LOCATE 16, 1 PRINT
9016 LOCATE 17, 1 PRINT
9017 LOCATE 18, 1 PRINT
9018 LOCATE 19, 1 PRINT
9019 LOCATE 20, 1 PRINT
9020 LOCATE 21, 1 PRINT
9021 LOCATE 22, 1 PRINT
9022 LOCATE 23, 1 PRINT
9023 LOCATE 24, 1 PRINT
9024 LOCATE 25, 1 PRINT
9025 LOCATE 26, 1 PRINT
9026 LOCATE 27, 1 PRINT
9027 LOCATE 28, 1 PRINT
9028 LOCATE 29, 1 PRINT
9029 LOCATE 30, 1 PRINT
9030 LOCATE 31, 1 PRINT
9031 LOCATE 32, 1 PRINT
9032 LOCATE 33, 1 PRINT
9033 LOCATE 34, 1 PRINT
9034 LOCATE 35, 1 PRINT
9035 LOCATE 36, 1 PRINT
9036 LOCATE 37, 1 PRINT
9037 LOCATE 38, 1 PRINT
9038 LOCATE 39, 1 PRINT
9039 LOCATE 40, 1 PRINT
9040 LOCATE 41, 1 PRINT
9041 LOCATE 42, 1 PRINT
9042 LOCATE 43, 1 PRINT
9043 LOCATE 44, 1 PRINT
9044 LOCATE 45, 1 PRINT
9045 LOCATE 46, 1 PRINT
9046 LOCATE 47, 1 PRINT
9047 LOCATE 48, 1 PRINT
9048 LOCATE 49, 1 PRINT
9049 LOCATE 50, 1 PRINT
9050 LOCATE 51, 1 PRINT
9051 LOCATE 52, 1 PRINT
9052 LOCATE 53, 1 PRINT
9053 LOCATE 54, 1 PRINT
9054 LOCATE 55, 1 PRINT
9055 LOCATE 56, 1 PRINT
9056 LOCATE 57, 1 PRINT
9057 LOCATE 58, 1 PRINT
9058 LOCATE 59, 1 PRINT
9059 LOCATE 60, 1 PRINT
9060 LOCATE 61, 1 PRINT
9061 LOCATE 62, 1 PRINT
9062 LOCATE 63, 1 PRINT
9063 LOCATE 64, 1 PRINT
9064 LOCATE 65, 1 PRINT
9065 LOCATE 66, 1 PRINT
9066 LOCATE 67, 1 PRINT
9067 LOCATE 68, 1 PRINT
9068 LOCATE 69, 1 PRINT
9069 LOCATE 70, 1 PRINT
9070 LOCATE 71, 1 PRINT
9071 LOCATE 72, 1 PRINT
9072 LOCATE 73, 1 PRINT
9073 LOCATE 74, 1 PRINT
9074 LOCATE 75, 1 PRINT
9075 LOCATE 76, 1 PRINT
9076 LOCATE 77, 1 PRINT
9077 LOCATE 78, 1 PRINT
9078 LOCATE 79, 1 PRINT
9079 LOCATE 80, 1 PRINT
9080 LOCATE 81, 1 PRINT
9081 LOCATE 82, 1 PRINT
9082 LOCATE 83, 1 PRINT
9083 LOCATE 84, 1 PRINT
9084 LOCATE 85, 1 PRINT
9085 LOCATE 86, 1 PRINT
9086 LOCATE 87, 1 PRINT
9087 LOCATE 88, 1 PRINT
9088 LOCATE 89, 1 PRINT
9089 LOCATE 90, 1 PRINT
9090 LOCATE 91, 1 PRINT
9091 LOCATE 92, 1 PRINT
9092 LOCATE 93, 1 PRINT
9093 LOCATE 94, 1 PRINT
9094 LOCATE 95, 1 PRINT
9095 LOCATE 96, 1 PRINT
9096 LOCATE 97, 1 PRINT
9097 LOCATE 98, 1 PRINT
9098 LOCATE 99, 1 PRINT
9099 LOCATE 100, 1 PRINT

Designing screens is the hard part. Coding screens is time consuming, boring, unchallenging, in a word, "the pits". With **ScreenCODE**, you can quickly and easily design your screen. Then with the push of a button it will write the code in any language--C, dBase II, III, III+, Basic, or Turbo Pascal.

With **ScreenSNAP** you can capture screens that you like from other programs, then with **ScreenCODE** modify them, and it will write the code to be used in your program. There's no reason to reinvent the wheel.

SlideSHOW is easier than Bricklin and will let you build self running demos or prototypes.

You get **ALL THREE** in one package. Order before December 31 1987, and we will give you a **FREE** disk of our most popular screens. Ready to modify or use as is.

Now through the end of the year, get all three for **\$149.00** (Shipping and handling included.)

To order call **1(800) 223-3419**

Centurion Systems 2724 West Waco Dr., Waco, TX 76710

Accept Visa, MasterCard, Check, Money Order, C.O.D.

30 DAY MONEY BACK GUARANTEE

CIRCLE 215 ON READER SERVICE CARD

The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's startup code checks the system's resources and uses whatever is available.

still
\$59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Mark Williams C 3.1 and 4.0
- Lattice C 3.2
- Turbo Pascal 3.0 and 4.0
- Logitech Modula-2/86

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add \$4.00 for shipping and handling.

The Tool Makers

P.O. Box 8976
Moscow, Idaho 83843
208-883-4979

CIRCLE 216 ON READER SERVICE CARD

OF INTEREST

(continued from page 154)

133/133a Hill Ln.
Shirley, Southampton
England SO1 5AF
0703-631441

Hardware

Lloyd I/O is now shipping its OMEGA MC68020-based workstation. The workstation provides integral floating-point math support via the MC68881 math coprocessor. It also includes 1 megabyte of zero-wait-state; nonvolatile static RAM; the OS-9, 68K, real-time, multitasking operating system; and a C compiler. The standard system configuration supports up to four users; further users are supported by optional I/O expansion boards. The base price for the workstation is \$4,750. Special system configurations, higher clock speeds, and hardware options are available for an additional cost. Reader Service No. 23.

Lloyd I/O Inc.
19535 Northeast Glisan St.
P.O. Box 30945
Portland, OR 97230
(800) 227-3719

An 80386-based multiuser system running the Xenix System V operating system, the Altos 386 Series 2000, is available from **Altos Computer Systems**. The Series 2000's design features modular architecture that allows memory and storage to be easily expanded. It also includes an intelligent file processor subsystem; system memory; a communications processor; an ESDI hard-disk drive; a 1.6 megabyte, 5¹/₄-inch floppy-disk drive; a 60-megabyte streaming magnetic tape unit; and an Altos V terminal. The 80386 processor runs at 16 MHz with a 32K data and instruction cache, an intelligent file processor, and an 80387 floating-point coprocessor.

The Altos 386 Series 2000 is available in four configurations ranging in price from \$25,000 to \$30,000. Reader Service No. 24.
Altos Computer Systems
2641 Orchard Pkwy.
San Jose, CA 95134
(408) 946-6700

DDJ

Upgrade your technology

The software technology available to programmers of IBM-compatible personal computers is truly amazing. And newer, more powerful development packages appear all the time. But until now, finding out about these important products has been a difficult and time consuming task.

FREE Buyer's Guide. The New Programmer's Connection Buyers Guide contains individual descriptions of over 500 titles of programmer's development software by over 150 manufacturers. Each description covers major product features as well as any software or hardware requirements and version numbers. In the box on the right are some examples of the types of descriptions you'll find in our Buyer's Guide.

No Hidden Charges. The low discount prices in our Buyer's Guide are all you pay. We don't charge extra for domestic UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours.

Noncommissioned Staff. Our courteous salespeople are always ready to help you. And if you aren't sure about your needs, our knowledgeable technical people can give you sound, objective advice.

Experience. We've specialized in development software for IBM-compatible personal computers since 1984 and

are experienced at providing a full range of quality products and customer services.

How to Get Your Copy. There are three ways for you to receive your FREE copy of the Programmer's Connection Buyer's Guide: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your Programmer's Connection Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

Announcing Our Super Year-end SALE

on selected Microsoft Products
until December 31, 1987

New Version 5.0 Microsoft C Optimizing Compiler 5.0

List \$450 Ours \$269
Sale Price \$259*

The Microsoft C Optimizing Compiler is a high performance, industry-standard C language development system. It uses many state-of-the-art optimizing techniques including: in-line code generation; loop invariant expression removal; automatic register allocation of variables; elimination of common subexpressions; improved constant folding & value propagation; and efficient coding techniques covered in the documentation. This extensive package also includes the new Microsoft QuickC Compiler (see separate description) that lets you quickly develop programs with its integrated environment and extremely fast compilation speeds. Microsoft C features: library routines that implement UNIX System V and most of the proposed ANSI standard libraries; extremely fast and optimized code generation; five memory models including HUGE; mixed models using NEAR, FAR and HUGE pointers; huge programs up to one MB and huge arrays larger than 64K; and source and object compatibility with XENIX. The package also includes Microsoft CodeView, an advanced source level debugger that features: support for overlays created with Microsoft Link; Expanded Memory Specification (EMS) support; access to source level and symbolic debug information from your Microsoft C, FORTRAN and MASM programs; simultaneous view of source code and assembly; display of local and global variables and expressions; animation and single-step; breakpoints and watchpoints; code and output screens; display of CPU registers and flags; and support for keyboard or optional mouse.

Requires 384K memory. Hard disk recommended. Version 5.0.

New Version 5.0 Microsoft Macro Assembler

List \$150 Ours \$93
Sale Price \$89*

The Microsoft Macro Assembler (MASM) is the fastest and most complete assembler development system ever created. Written in Microsoft C, MASM assembles programs at blinding speeds. Make your programs run faster by linking assembly language subroutines to your Microsoft QuickBASIC, C, QuickC, FORTRAN or Pascal programs. A broad range of examples in the completely revised and expanded documentation makes it easy to write assembly language routines, plus template programs are on-line so all you have to do is plug in your code and go — MASM takes care of the interfacing. There is also a complete reference to the instruction set containing extensive examples. The Macro Assembler itself has sped up by 25-40% and the linker has doubled its speed since version 4.0. Now there is support for the 80386 and 80387 instruction set. Microsoft CodeView, the window-oriented, source-level debugger gives you total control over debugging. This highly acclaimed debugger is ideal for debugging your assembly language subroutines called from QuickBASIC, C, QuickC, Pascal or FORTRAN. Debug using your original source code and comments. Watch the value of variables and expressions change while your program executes and view registers and flags.

Requires 256K memory. Version 5.0.

*Plus a Special Rebate from Microsoft

Purchase any two of the following products before January 31, 1988 and receive an additional \$30 rebate directly from Microsoft. Or purchase any three and receive a \$50 rebate:

- Microsoft C Optimizing Compiler
- Microsoft Macro Assembler
- Microsoft QuickC
- Microsoft QuickBASIC

CALL TOLL FREE

USA: 800-336-1166

Canada: 800-225-1166

Ohio & Alaska

(Collect): .. 216-494-3781

International: 216-494-3781

Telex: 9102406879

Easylink: 62806530

Programmer's Connection
7249 Whipple Avenue NW
North Canton, OH 44720

Please turn the page for our latest price list and ordering information. ➡



The Free Programmer's Connection Buyer's Guide.

ai - expert systems

1st-CLASS by Programs in Motion	495	399
EXSYS Development Software by EXSYS	395	289
EXSYS Runtime System	600	469
LEVELS by Information Builders	685	569
Logic-Line Series All varieties by Thunderstone	CALL	CALL
VP EXPERT by Paperback Software	100	63

ai - lisp language

muLISP-87 Interpreter by Soft Warehouse	300	199
muLISP-87 Interpreter & Compiler	400	269
Q'Nial Various by NIAL Systems	CALL	CALL
Star Sapphire LISP Compiler by Sapiens	495	429
TransLISP PLUS from Solution Systems	195	125

ai - prolog language

Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG All Varieties	CALL	CALL
MPROLOG Language Primer LOGICWARE	50	45
MPROLOG P500 by LOGICWARE	495	395
MPROLOG P550 w/Primer by LOGICWARE	220	175
Turbo PROLOG by Borland Intl	100	64
Turbo PROLOG Toolbox by Borland Intl	100	64

ai - smalltalk language

Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette	50	45
Smalltalk/Comm	50	45

ai - texas instruments

Arborist Decision Tree Software	595	519
PC Scheme Lisp	95	77
Personal Consultant Easy	495	435
Personal Consultant Plus	495	435
Personal Consultant Online	995	869
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

ada language

AdaVantage GSA-Validated by Meridian Software	795	735
AdaVantage Utility Packages	50	47
DOS Environment Package	50	47
Janus/ADA C Pak by R&R Software	95	84
Janus/ADA D Pak by R&R Software	1250	1059
Janus/ADA ED Pak by R&R Software	395	349

apl language

APL+PLUS PC by STSC	695	495
APL+PLUS PC Spreadsheet Mgr by STSC	195	139
APL+PLUS PC Tools by STSC	CALL	CALL
APL+PLUS PS/2 by STSC	695	495
ATLAS GRAPHICS by STSC	450	329
Financial/Statistical Library by STSC	275	189
Pocket APL by STSC	95	69
STATGRAPHICS by STSC	895	649

assembly language

386 ASM/LINK Cross Asm by Phar Lap	495	389
8088 Assembler w/2-80 Translator by 2500 AD	100	89
ASMLIB Function Library by BCSoft	149	125
asmTREE B-Tree Dev System by BCSoft	395	329
Cross Assemblers Various by 2500 AD	CALL	CALL
EZASM by C Source	70	59
Microsoft Macro Assembler	150	89
Turbo Debugger by Speedware	89	79
Turbo Editasm by Speedware	99	84
Visible Computer: 8088 by Software Masters	80	64

basic language

db/Lib for QuickBASIC by AJS Publishing	139	119
Finally by Komputerwerk	99	85
MACH 2 by Micro Help	69	55
Microsoft QuickBASIC 4.0	99	59
QBase Relational Database by Crescent	89	79
Quick-Tools by BCSoft	130	109
QuickPak by Crescent Software	69	59
Scientific Subroutine Library by Wiley	125	99
Screen Sculptor by Software Bottling	125	91
Stay-Res by MicroHelp	69	55
True BASIC	100	79
True BASIC w/Run-time	200	99
True BASIC 3D Graphics	50	41
True BASIC Developer's Toolkit	50	41
Turbo BASIC Compiler by Borland Intl	100	64

blaise products

ASYNCH MANAGER Specify C or Pascal	175	135
C TOOLS PLUS/5.0	129	99
KeyPlayer Super Batch Program	50	45
LIGHT TOOLS for Datatight C	100	65
PASCAL TOOLS	125	95
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF Text Formatter	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER Specify C or Pascal	275	199

borland products

EUREKA Equation Solver	167	105
Reflex: The Analyst	150	99
Sidekick	85	57
Superkey	100	64
Turbo Basic Compiler	100	64
Turbo Basic Database Toolbox	100	64
Turbo Basic Editor Toolbox	100	64
Turbo Basic Telecom Toolbox	100	64
Turbo C Compiler (Call for support products)	100	64
Turbo Lightning and Word Wizard	150	94

Turbo Lightning	100	64
Turbo Lightning Word Wizard	70	47
Turbo Pascal and Tutor	CALL	CALL
Turbo Pascal	100	64
Turbo Pascal Tutor	70	41
Turbo Pascal Database Toolbox	100	64
Turbo Pascal Developer's Toolkit	395	289
Turbo Pascal Editor Toolbox	100	64
Turbo Pascal Gameworks Toolbox	100	64
Turbo Pascal Graphix Toolbox	100	64
Turbo Pascal Numerical Methods Toolbox	100	64
Turbo Prolog Compiler	100	64
Turbo Prolog Toolbox	100	64

c language

C-terp by Gimpel, Specify compiler	298	219
C Trainer with Book by Catalystix	122	87
DeSmet C w/Debugger & Large case	209	184
DeSmet C w/Debugger only	159	138
Eco-C88 Modeling Compiler by Ecosoft	100	79
Instant C by Rational Systems	495	369
Lattice C Compiler vers. 3.2 from Lattice	500	265
Mark Williams Let's C with FREE csd	75	54
Microsoft C Compiler 5.0 w/CodeView	450	259
Microsoft QuickC Compiler	99	59
Optimum-C by Datatight	139	95
Turbo C Compiler by Borland	100	64
Uniware 68000/10/20 Cross Compiler by SDS	995	829

c utilities

Blackstar C Library by Sterling Castle	125	98
C++ by Guidelines w/version 1.1 kernel	195	172
c-tree & r-tree Combo by FairCom	650	519
c-tree ISAM File Manager	395	315
r-tree Report Generator	295	239
Csharp Realtime Toolkit by Systems Guild	600	489
Curses Window Dev Pkg by Aspen Scientific	119	105
with Source Code	289	249
dBx dBASE to C Translator by Desktop AI	350	299
with Source Code	550	419
Flash-up by Software Bottling	90	78
Graphic/C color version by Sci Endeavors	350	274
HALO Graphics by Media Cybernetics	300	205
HALO Development Pkg for Microsoft	595	389
The HAMMER by QES Systems	195	129
PANEL/TC for Turbo C by Roundhill	129	95
PANEL Plus by Roundhill	495	395
PC Limit by Gimpel Software	139	99
RTC PLUS Fortran to C by Cobalt Blue	450	399
Sapiens V8 Virtual Memory Manager	300	265
Scientific Subroutine Library by Wiley	175	135
TE Text Editor source by Sub Systems	95	85
Vitamin C by Creative Programming	225	149
VC Screen Forms Designer	100	79
Zview by Data Mgmt Consultants	245	139

cobol language

COBOLspil by Flexus	395	329
FLPLIB for Realia COBOL by BCSoft	149	129
Micro Focus COBOL See Micro Focus Section		
Microsoft COBOL See Microsoft Section		
PCDT by Pro-Code	995	895
Realia COBOL with RealMENU	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL by Ryan-McFarland	950	639
RM/COBOL 85 by Ryan-McFarland	1250	895
RM/NET+5 by Ryan-McFarland	300	259
RM/Screens	395	334
SCREENIO by Norcam	400	379
screenplay for COBOL by Flexus	175	129

c++ products

Combo Package by Custom Software Systems	199	175
PC/TOOLS UNIX-like Utilities	49	45
PC/VI vi Editor	149	99

database management

Advanced DBMaster by Macon Software	510	419
dBASE III Plus by Ashton-Tate	695	389
dbIII Compiler by WordTech Systems	NEW	CALL
dbSQL by WordTech Systems	NEW	CALL
dbXL by WordTech Systems	NEW	169
dFLOW by Wallsoft	NEW	149
The Documenter by Wallsoft	NEW	295
Genifer by Bytel	NEW	395
Paradox 1.1 by Ansa/Borland	NEW	495
Paradox 2.0 by Ansa/Borland	NEW	725
Paradox Network Pack by Ansa/Borland	NEW	995
Q&A by Symantec	NEW	349
Quickcode PLUS by Fox & Geller	NEW	295
Quickindex by Fox & Geller	NEW	149
Quickreport by Fox & Geller	NEW	295
QuickSilver by WordTech Systems	NEW	599
R-Base 5000 by Microrim	NEW	495
R-Base System V by Microrim	NEW	700
\.rdb by Robinson-Shafer-Wright	NEW	139
Tom Rettig's Library by Tom Rettig & Assoc	NEW	100
UI Programmer by Wallsoft	NEW	295
VP INFO by Paperback Software	NEW	100

debuggers & profilers

386 DEBUG Cross Debugger by Phar Lap	195	129
Advanced Trace-86 by Morgan Computing	175	115
Codemith-86 by Visual Age	145	98
DSDB7 by Soft Advances	125	79
MiniProbe by Altron	395	369
Periscope I with Board by Periscope	345	275
Periscope II-X Software only	175	139
Periscope III 10 MHz version	145	105
Periscope III 8 MHz version	995	795
Periscope III 10 MHz version	1095	875
The PROFILER with Source Code by DWB	125	89
TURBOsmith Source debugger for Turbo Pascal	99	89
The WATCHER Profiler by Stony Brook	60	51

disk utilities

Back-It by Gazelle Systems	130	115
Disk Optimizer by Softlogic Systems	60	55

Disk Technician by Prime Solutions	100	89
FASTBACK by 5th Generation Systems	179	129
Take Two Manager United Software Security	139	119
Vcache by Golden Bow Systems	50	47
Vopt by Golden Bow Systems	50	47
Vfeature by Golden Bow Systems	80	74
Vfeature Deluxe by Golden Bow Systems	120	111
XenoCopy-PC by XenoSoft	80	69

dos utilities

Advanced Norton Utilities	150	89
Command Plus by ESP Software	80	69
Desqview from Quarterdeck	130	115
FANSI-CONSOLE by Hersey Micro	75	62
Mace Utilities Paul Mace Software	99	89
MicroHelp Utility by MicroHelp	59	49
Norton Commander by Peter Norton	75	55
Norton Utilities by Peter Norton	100	59
OPAL Shell Language by Software Factory	99	89
Q-DOS II by Gazelle Systems	70	59
Taskview by Sunny Hill Software	80	55

essential products

Breakout Debugger Only Any language	125	89
C Utility Library	185	119
Essential Communications	185	125
Essential Communications with Break Out	250	189
Essential Graphics	250	183
/resident _C_/	NEW	99
with Source Code	NEW	198
ScreenStar	NEW	99
with Library Source Code	NEW	198

forth language

CFORTH Native Code Compiler by LMI	300	229
FORTH/83 Metacompiler Specify Target	750	599
PC/FORTH by Laboratory Microsystems	150	109
PC/FORTH+ by Laboratory Microsystems	250	199
Programmer's Package #1 by LMI	250	199
Programmer's Package #2 by LMI	350	279
Programmer's Package #3 by LMI	500	399
UR/FORTH Also Available for OS/2 by LMI	350	279
UR/FORTH Libraries	500	395

fortran language

50 MORE: FORTRAN by Peerless Scientific	125	95
ACS Time Series by Alpha Computer Service	495	389
AUTOMATED PROGRAMMER by KGK Automated	995	949
Essential Graphics by Essential Software	250	183
Forlib-Plus by Alpha Computer Service	70	44
FORTRAN Addendum by Impulse Engr	95	85
FORTRAN Addenda by Impulse Engr	165	138
HALO Graphics by Media Cybernetics	300	205
I/O PRO w/No Limit Library by MEF	250	219
Microcompatibles Combo Package	240	215
Plumatic	135	117
Plumatic	135	117
Microsoft FORTRAN w/CodeView	450	259
No Limit Library by MEF Environmental	129	109
Numerical Analyst by MAGUS	295	249
PANEL by Roundhill Computer Systems	295	199
RM/FORTRAN by Ryan-McFarland	599	399
RTC PLUS Fortran to C by Cobalt Blue	450	399
Scientific Subroutine Library by Wiley	175	135
Statistician by Alpha Computer Service	295	235
STATLIB.GL by Wiley	295	239
STATLIB.TSF by Wiley	295	239
Strings & Strings by Alpha Computer Service	70	45

greenleaf products

Greenleaf C Sampler for QuickC & Turbo C	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Library	225	155
with Source Code	395	249
Greenleaf Functions	185	125

help utilities

HELP/Control by MDS	125	99
On-line Help from Opt-Tech	149	99
SoftScreen/HELP by Dialectic Systems	195	149

lattice products

Lattice C Compiler ver 3.2 from Lattice	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord Function Library	150	95
with Source Code	300	179
C-Sprite Source Level Debugger	175	CALL
Curses Screen Manager	125	85
with Source Code	250	169
dBc III	250	169
with Source Code	500	356
dBc III Plus	750	594
with Source Code	1500	1184
LMK Make Facility	195	138
RPg II Combo All three items below	1100	875
RPg II Compiler No Royalties	750	625
SEU Source Entry Utility	250	199
Sort/Merge	250	199
Screen Design Aid Utility for RPg II	350	309
SecretDisk II Encryption Utility	120	88
SideTalk Resident Communications	120	88
SXP/PC Scientific Subroutine Library	350	269
Text Management Utilities	120	88

metagraphics products

LightWINDOW/C for Datatight C	95	79
MetaWINDOW No Royalties	195	159
MetaWINDOW PLUS	275	229
TurboWINDOW/C for Turbo C	95	79
TurboWINDOW/Pascal for Turbo Pascal	95	79

micro focus products

Micro Focus COBOL/2	900	729
Micro Focus Level II COBOL w/Animator	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX	CALL	CALL

Micro Focus PC-CICS	New	1495	1189
with Micro/SPF	New	1595	1269
Micro Focus Personal COBOL		149	119
Micro Focus Professional COBOL		2000	1595
Micro Focus VS COBOL/XENIX		1495	1195
Micro Focus Support Products:			
COBOL/IO Ad hoc Report Writer		495	395
COBOL/IO for DOS 3.X Networks		995	795
FORMS-2		295	235
SOURCEWRITER		995	795

microport products

386 Unlimited License Kit		249	209
AT Unlimited License Kit		249	209
DOSMerge286 Run DOS and UNIX together		149	129
DOSMerge386 Run DOS and UNIX together	CALL	CALL	CALL
System V/386 Combination		799	669
386 Runtime System		199	169
386 Software Development System		499	429
Text Preparation System		199	169
System V/AT Combination		549	465
AT Runtime System		199	169
AT Software Development System		249	209
Text Preparation System		199	169

microsoft products

Microsoft BASIC Compiler for XENIX		695	419
Microsoft BASIC Interpreter for XENIX		350	209
Microsoft C Compiler 5.0 w/CodeView	SALE	450	259
Microsoft COBOL Compiler with COBOL Tools	SALE	700	419
for XENIX	SALE	995	599
Microsoft Excel	New	495	295
Microsoft FORTRAN Optimizing Compiler	SALE	450	259
Microsoft FORTRAN for XENIX		695	409
Microsoft Learning DOS		50	36
Microsoft Macro Assembler	SALE	150	89
Microsoft Mouse Specify Serial or Bus		150	99
with Microsoft Windows		200	135
with EasyCAD		175	114
Microsoft Pascal Compiler	SALE	300	174
for XENIX	SALE	695	409
Microsoft QuickBASIC 4.0	SALE	99	59
Microsoft QuickC	SALE	99	59
Microsoft Windows		99	63
Microsoft Windows Development Kit		500	299
Microsoft Word		450	269
Microsoft Works	New	195	119

mks products

MKS AWK		75	65
MKS RCS Revision Control System	New	189	155
MKS Toolkit with MKS VI Editor		139	109
MKS Trilogy with AWK, CRYPT & Korn Shell	New	119	99
MKS VI Editor by MKS		75	65

modula-2 language

LOGITECH Modula-2 Development System		249	199
Modula-2 Compiler Pack		99	79
Modula-2 Toolkit		169	139
LOGITECH Modula-2 ROM Pkg		299	239
LOGITECH Modula-2 Window Pkg		49	39
Macro2 Macro preprocessor by PMI		89	79
ModBase by PMI		89	79
Repertoire by PMI		89	75
Science & Engrg Tools by Quinn-Curtis		75	67
Universal Graphics Library by Quinn-Curtis		130	119

mouse products

LOGIMOUSE BUS with PLUS Pkg by LOGITECH		119	98
with PLUS & PC Paintbrush		149	119
with PLUS & CAD Software		189	153
with PLUS & CAD & Paint		219	179
with PLUS & First Publisher	CALL	CALL	CALL
LOGIMOUSE C7 with PLUS Pkg, Specify Connector		119	98
with PLUS & PC Paintbrush		149	119
with PLUS & CAD Software		189	153
with PLUS & CAD & Paint		219	179
with PLUS & First Publisher	CALL	CALL	CALL
Microsoft Mouse See Microsoft Section			

other languages

ACTOR by Whitewater Group		495	419
CCS MUMPS All varieties by MGlobal	CALL	CALL	CALL
Marshall Pascal by Marshall Language Systems		189	155
Pascal-2 by Oregon Software		395	325
Personal REXX by Mansfield Software		125	99
SNBOL4+ by Catspaw		95	80

other products

Carbon Copy Plus by Meridian Technology		195	159
Dan Bricklin's Demo Pgm by Software Garden		75	57
Dan Bricklin's Demo Tutorial		50	45
Fast Forward by Mark Williams		70	59
Instant Replay by Nostradamus		150	CALL
MicroTeX Typesetting from Addison-Wesley		295	CALL
Printer Drivers	New	CALL	CALL
nuMATH by Soft Warehouse	New	300	199
Net-Tools by BCSoft		149	129
Norton Guides Specify Language		100	65
OPT-Text Sort by Opt-Tech Data Proc		149	99
PC/TOOLS by Custom Software		49	45
Resident Expert Specify lang by Santa Rita	New	59	55
Screen Machine by MicroHelp		79	59
SuperSort by LifeStyle		139	119

phoenix products

Pasm86 Macro Assembler version 2.0		195	108
Pdisk Hard Disk & Backup Utility		145	99
Plantasy Pac Phoenix Combo		595	535
Plinity Execution Profiler		395	209
Plix86plus Symbolic Debugger		395	209
PforCa Specify C Compiler		395	209
PforCa++ Specify C Compiler and C++		395	209
Plink86plus Overlay Linker		495	275
Pmaker Make Utility		125	78
Pmate Macro Text Editor		195	108
Pre-C Lint Utility		295	154
Ptel Binary File Transfer Program		195	108

polytron products

PolyBoost Software Accelerator	Special Price	80	54
PolyDesk III		99	72
PolyDesk III Archivist		50	42
PolyDesk III Cryptographer		50	42
PolyDesk III Talk		70	52
PolyLibrarian Library Manager		99	89
PolyLibrarian II Library Manager		149	129
PolyMake UNIX-like Make Facility		149	129
PolyShell	Special Price	149	105
Polytron C Beautifier		50	45
PolyXREF Complete Cross Ref Utility		219	185
PolyXREF One language only		129	109
PVCS Corporate Version Control System		395	329
PVCS Personal		149	129

program mgmt utilities

Interactive EASYFLOW by Haventree	New Version	150	125
PrintQ by Software Directions		89	84
Quilt Computing Combo QMake & SRMS		250	199
Sapiens MAKE		179	155
Sapiens MAKE & V8		439	379
Source Print by Aldebaran Labs		97	75
TLIB Version Control System by Burton		100	89
Tree Diagrammer by Aldebaran Labs		77	67

sco products

Complete XENIX System V by SCO		1295	994
Development System		595	499
Operating System Specify XT or AT		595	499
Text Processing Package		195	144
Lyrinx by SCO		595	499
SCO Professional 1-2-3 Worklike for XENIX		795	595
SCO XENIX-NET		595	495
XENIX System V 386 by SCO	CALL	CALL	CALL

softcraft products

Btrieve ISAM Mgr with No Royalties		245	184
Xtrieve Query Utility		245	184
Report Option for Xtrieve		145	99
Btrieve/N for Networks		595	454
Xtrieve/N		595	454
Report Option/N for Xtrieve/N		345	269
XQL	New	795	CALL

text editors

Brief & dBrief Combo from Solution Systems		275	CALL
Brief		195	CALL
dBrief Customizes Brief for dBASE III		95	CALL
de by David Livshin		75	65
Epsilon Emacs-like editor by Lugaru		195	147
KEDIT by Mansfield Software		125	98
Micro/SPF by PHASER SYSTEMS		175	139
Microsoft Word		450	269
PC/VI by Custom Software Systems		149	99
SPF/PC by Command Technology Corp	CALL	CALL	CALL
Vedit Plus by CompuView		185	128

turbo pascal utilities

ALICE Interpreter by Software Channels		95	66
AZATAR DOS Toolkit by AZATAR	New	95	85
DOS/BIOS & Mouse Tools by Quinn-Curtis		75	67
Flash-up by Software Bottling	New	89	78
Flash-up Developer's Toolbox	New	49	45
MACH 2 for Turbo Pascal by Micro Help		69	55
MetaByte D/A Tools by Quinn-Curtis		100	89
Science & Engrg Tools by Quinn-Curtis		75	67
Screen Scriptor by Software Bottling		125	91
Speed Screen by Software Bottling		35	32
System Builder by Royal American	New Version	150	129
IMPEX Query Utility		100	89
Report Builder	New Version	130	115
TDebugPLUS by TurboPower Software		60	49
Tmark by Tangent Designs		80	69
Turbo EXTENDER by TurboPower Software		85	64
Turbo OPTIMIZER by TurboPower		75	65
with Source Code		125	98
Turbo Professional by Sunny Hill		70	45
TurboHALO from IMSI		95	75
TurboPower Utilities by TurboPower		95	78
TurboRef by Gracon Services		50	35
TURBOSmith Source Debugger by Visual Age		99	89
Universal Graphics Library by Quinn-Curtis		130	119

wendin products

Operating System Toolbox		99	79
PCNX Operating system		99	79
PCVMS Similar to VAX/VMS		99	79
Wendin-DOS Multitasking DOS		99	85
Wendin-DOS Application Developer's Kit		99	85
XTC Text Editor w/Pascal source		99	75

xenix/unix products

Btrieve ISAM File Mgr by SoftCraft		595	454
C-terp by Gimpel, Specify compiler		498	379
c-tree ISAM Mgr by FairCom		395	315
dbx with Library Source by Desktop AI		550	419
DIRECTORY SHELL 286 by American Mgmt Sys		349	295
DIRECTORY SHELL 386 by American Mgmt Sys	New	495	415
DOSIX Console Version by Data Basics		400	349
DOSIX User Version by Data Basics		200	179
Lugaru Text Editor by Epsilon	New	195	147
Micro Focus Products See Micro Focus Section			
Microport Products See Microport Section			
Microsoft Products See Microsoft Section			
PANEL Plus by Roundhill Computer Systems		795	535
REAL-TOOLS Binary Version by PCT		89	89
Library Source Version		599	539
Complete Source Version		999	723
RM/COBOL by Ryan-McFarland		1250	948
RM/FORTRAN by Ryan-McFarland		750	549
SCO Products See SCO Section			

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of \$100) are available to qualified US accounts only.

SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional \$10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection

7249 Whipple Ave. NW

North Canton, OH 44720

USA	800-336-1166
CANADA	800-225-1166
OHIO & ALASKA (Collect)	216-494-3781
TELEX	9102406879
EASYLINK	62806530

INTERNATIONAL	216-494-3781
CUSTOMER SERVICE	216-494-8899

Hours: Weekdays 8:30 AM to 8:00 PM EST.



Terms are subject to change.
©1987 Programmers Connection, Inc.

CIRCLE 217 ON READER SERVICE CARD

SWAINE'S FLAMES

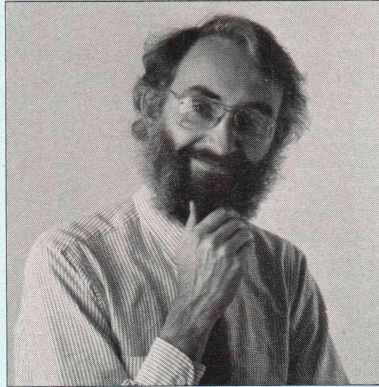
Remember the good old days when ordinary people were afraid of computers? Then high tech became trendy. I wouldn't be surprised if Helen Gurley Brown's Cosmopolitan editorial this month began, "I just love operating systems, don't you? They're so earthy and fundamental. And 1987 was such an OS-some year!"

And Helen would be right; 1987 was OS-some. Microsoft and IBM revealed the master plan for the timed release of OS/2, the operating system for the 1990s, complete with developer's kits and seminars, and began delivering the pieces right on schedule.

Then, just when you thought that DOS was a stable development environment (OS-sified?), Microsoft and a cast of thousands announced (1) the Lotus/Intel/Microsoft Expanded Memory Specification (give a point to Quarterdeck) and (2) Windows/386 (take it away again). Now, modulo a few fatal bugs that would undoubtedly be fixed in the next release, you could stick code as well as data up above 640K, run more and bigger TSRs, switch instantaneously among existing applications; here was multitasking before OS/2.

It wasn't immediately obvious just what effect the release of Windows/386 would have on the reception of OS/2. It did seem that the peculiar benefits of OS/2 would be realized only with the arrival of OS/2-specific software, emphasizing inter-application communication or intra-application multitasking. And it depressed the Microsoft OS/2 developers, who longed to forget the 286 processor and get on with the real thing, OS/2/386, before Intel released the 486 and really depressed them.

While we were all waiting for OS/2, a few companies went ahead with 386 tools, including Phar Lap, MetaWare, Softguard, and The Software



Link. Compaq offered a version of DOS that eliminated DOS's disk-file size limitations. And Digital Research delivered Concurrent DOS 386, a DOS 3.3-compatible multiuser, multitasking windowing operating system for 386 machines.

In that other universe, Apple released Multifinder, allowing users to switch smoothly between existing Mac applications. But wait; there's more.

Meanwhile, a number of developments brought Unix into prominence as an operating system for personal computers. There was the narrowing power gap between personal computers and workstations (and the entrenchment of Unix on engineering workstations). There were the steps toward standardization seen in X/Open and IEEE Posix, and the movement toward convergence of the Berkeley and AT&T versions. Although they flaunted standardization efforts, you wouldn't want to discount IBM's endorsement of Unix with its AIX (to be available on the RT, PS/2, and System 370), and Apple's ditto with A/UX. Nor would you want to discount the influence of Unix supporters Sun and NeXT, either, or all those sex, drugs, and Unix buttons at the Hackers' Conference. Then there was the growing support for the Unix graphical interface, Xwindows.

With the major operating systems all moving toward some form of multitasking and all providing some sort of windowing user interface, what the world was coming to need,

many believed, were tools to ease the development process for window-oriented software, and to make it easier to port code across windowing environments. The White-water Group did well pitching its object-oriented language, Actor, as a tool to make Windows development easier, and began work on their Mac version.

Michael Bentley saw the same need. While reviewers gushed over Danny Goodman's book on Bill Atkinson's Hypercard (a thorough and readable book about an interesting product), Michael Bentley's *The Viewport Technician*, which promised not only to show how to develop code portable across windowing environments, but how to make that portable code efficient, got little press. Did the book deliver on its promise? Only someone who had developed code for the Amiga, GEM, Windows, and the Mac could say.

Odds and ends:

Contrary to Allen Holub's expectations, Microsoft did not deliver Quick C when promised, any more than Borland delivered Turbo C when promised.

The third edition of Daniel Remer and Stephen Elias's *Legal Care for Your Software* is now out. I'm not convinced that the authors have done their job with respect to the tricky area of ownership of reusable code in work done for hire, but generally this is a good first book in the legal issues in software development and sale. It's published by Nolo Press, 950 Parker Street, Berkeley, CA 94710.

Michael Swaine

Michael Swaine
editor-in-chief

How A C Programmer Became A Screen Star

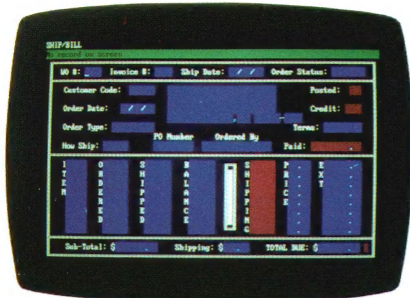
Screens, the Visible Part of Your Program.

A program is often judged by how well the screens are executed. However, the real creativity lies in what goes on behind the screens.

ScreenStar is a product that allows your real creativity to light up the screen. It reduces costly screen, window, and data validation development time.

You Take the Bows, We Write the Code.

Our natural drawing commands allow you to paint any screen imaginable. Press one key when you are satisfied and ScreenStar produces concise, commented, ready-to-compile code. This allows immediate testing of the I/O screens, including smooth, even scrolling between multiple screens.



Create or capture complex screens with data-entry filters built in.

If all ScreenStar did was turn screens into code it would be a useful tool. Yet ScreenStar also permits a wide range of field types. Some of the choices include date, alphanumeric, telephone, yes/no, dollar, time and user-definable fields.

Other valuable data-entry filters are built in, such as required field, display only, and many others. All screen fields are generated with error-checking routines.

ScreenStar Not Only Captures Your Imagination, It Captures Screens.

The memory-resident capture program converts any screen into a ScreenStar file in seconds, including those generated by programs like Dan Bricklin's Demo Program.

ScreenStar Sets the Stage for Windows.

ScreenStar comes with a complete window generating library. You design the help screens and pop-up windows. Essential ScreenStar windowing functions tie them together in one smooth package.

Curtain Call.

They may not ask for your autograph, but they will want to know how you did those screens. Screenstar is more than a screen-painting program. It is a screen processor. No professional programming environment will be complete without this product.

We know you will enjoy using ScreenStar. However, should you give it less than rave reviews, return it within 30 days for a full refund.

★ Interactive screen painting and subsequent code generation.

★ Multiple screen design and scrolling.

★ TSR screen capture program, works with any program including Dan Bricklin's Demo Program.

★ Complete window design including overlapping window functions.

★ Screens are compressed into data structures, and remain a permanent part of the program. No messy data files to look for.

Price - \$99

W/Source add \$99

**Audition Our Product
Today. Call:**

(201) 762-6965

Essential Software
South Orange Plaza
76 South Orange Ave., Suite 3
South Orange, NJ 07079

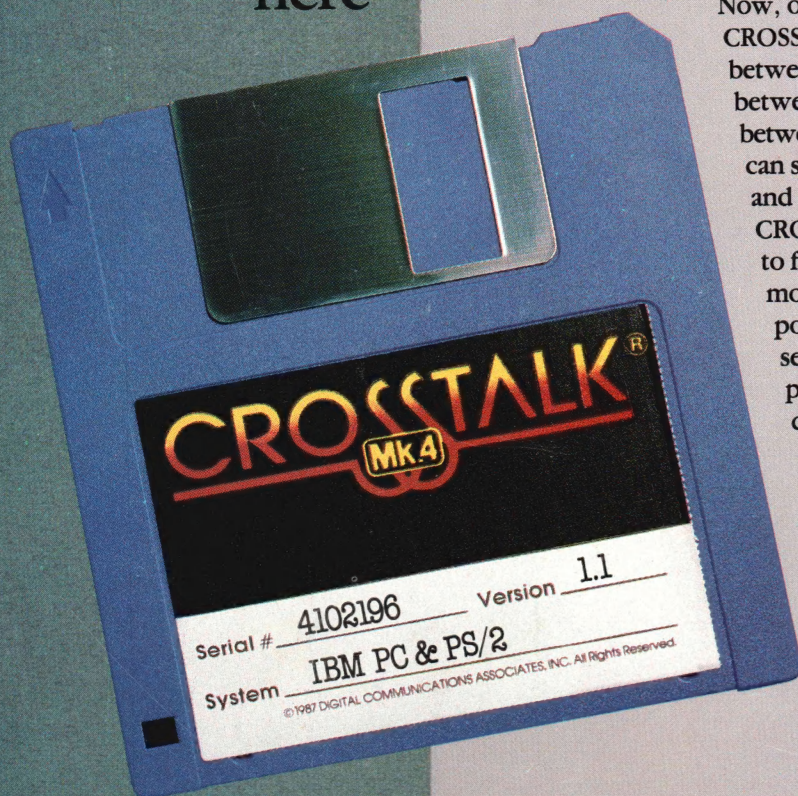
CIRCLE 218 ON READER SERVICE CARD

IBM
Spoken
Here

and
here

and
here

and
here



Whatever dialect of IBM you need to speak, CROSSTALK® Mk. 4 makes the connection.

Now, one program does the job that used to require several. CROSSTALK® Mk. 4 allows high-speed direct communications between PCs and minicomputers, or (with an IRMA™ board) between your PC and an IBM Mainframe, or (with Smart Alec™) between your PC and IBM System 3x's. If you like, CROSSTALK can support all of these sessions (and others) simultaneously, and display each session in its own window.

CROSSTALK Mk. 4 emulates all the terminals you're likely to find useful. That includes IBM 3101 (page and character modes), IBM 525x, IBM 529x, IBM 327x, as well as many popular async terminals like the DEC VT100 and VT220 series. CROSSTALK Mk. 4 includes the powerful CASL™ programming language, which allows you to automate communications applications quickly and easily.

So if you're used to thinking of CROSSTALK just to use with a modem, you're missing some important connections. Ask your dealer for details, or write:

dca Digital Communications Associates, Inc.
1000 Holcomb Woods Parkway / Roswell, Georgia 30076
1-800-241-6393

CROSSTALK®
COMMUNICATIONS

CROSSTALK and DCA are registered trademarks of Digital Communications Associates, Inc. IRMA, Smart Alec and CASL are trademarks of Digital Communications Associates, Inc. IBM is a registered trademark of International Business Machines Corp. DEC is a registered trademark of Digital Equipment Corp.